UNIVERSIDAD DE SEVILLA

Escuela Superior de Ingeniería

Contribuciones al Diseño de Arquitecturas de Sistemas Distribuidos Abiertos para la Provisión de Servicios del Cuidado de la Salud y de Soporte a la Autonomía del Ciudadano

Tesis Doctoral

Jorge Calvillo Arbizu

Ingeniero de Telecomunicación

Sevilla, 2012

Grupo de Ingeniería Biomédica, CIBER-BBN

Escuela Superior de Ingeniería

UNIVERSIDAD DE SEVILLA

Contribuciones al Diseño de Arquitecturas de Sistemas Distribuidos Abiertos para la Provisión de Servicios del Cuidado de la Salud y de Soporte a la Autonomía del Ciudadano

Tesis Doctoral

Autor: **Jorge Calvillo Arbizu**

Directoras: **Isabel Román Martínez, Laura María Roa Romero**

Sevilla, 2012

Agradecimientos

Este trabajo de investigación y su culminación como Tesis Doctoral ha sido una tarea ardua en algunos momentos, desesperante en otros y satisfactoria en la mayoría de ellos. Un trabajo individual que no obstante se ha visto influenciado por todas aquellas personas que en mayor o menor medida me han acompañado en esta etapa de mi vida, hayan sido unos pocos meses o los cinco años que se cumplen con la versión final del documento. Dar las gracias a todos ellos es una labor que pretendo hacer en persona pero queden aquí reflejados de forma general mis más sinceros agradecimientos.

A mi familia por haber respetado mis decisiones y el camino elegido aunque pudieran no estar completamente de acuerdo. Por haber respetado mi espacio y haberme dado su apoyo en estos años tan difíciles para todos. A mis amigos por el apoyo incondicional, las muestras de cariño e incluso las bromas a costa de la tesis. Por haber estado ahí para escuchar todo lo bueno y lo malo que tenía que contar sobre el proceso y por sufrirme día a día.

A todos los miembros del Grupo de Ingeniería Biomédica tanto presentes como pasados. Todos ellos han acompañado mi inicio investigador y he aprendido cosas de cada uno de ellos, no sólo en la dimensión investigadora sino también en la humana. Gracias en especial a Laura por haberme enseñado tanto en estos años y haber apostado por mí. Por enseñarme especialmente a no perder la coherencia y a ser persona por encima de investigador. También por compartir y poner en valor mis logros.

Y gracias por encima de todos a Isabel, no sólo por haberme guiado en el duro camino de la investigación sino por haber sido desde el primer momento un apoyo, una referencia y una amiga con la que he podido hablar en los buenos y en los malos momentos.

Y por último gracias a todas las personas anónimas del mundo científico que, sin saberlo, han despertado mi interés por la investigación, han avivado mi curiosidad y retado mi afán de superación.

A mi padre

Índice

Índice de Figuras

Indices de Tablas

Resumen

Resumen

El dominio sanitario se ha visto enormemente influenciado por las Tecnologías de la Información y la Comunicación (TICs) en las últimas décadas. Consecuencia de esto son los avances en dispositivos, gestión de información y procesos asistenciales. En parte debido a la heterogeneidad tecnológica de las soluciones y a la ausencia de una metodología formal de aplicación de las TICs, en la actualidad el escenario sanitario está fragmentado en sistemas separados que rara vez cooperan entre sí para proveer capacidades avanzadas. Esto dificulta la mejora en la eficiencia de los procesos, la evolución del sistema sanitario y la reducción de costes al existir soluciones redundantes que en ocasiones coexisten en las organizaciones sanitarias. La práctica clínica también está evolucionando hacia escenarios descentralizados donde la asistencia a un individuo es compartida entre diferentes organizaciones sanitarias (en ocasiones incluso distribuidas geográficamente) y en los que la coherencia de la información así como su privacidad son requisitos indispensables para una mejora de la eficiencia. En este escenario descentralizado los individuos y no las organizaciones deben ser el centro de los procesos, sustituyendo el actual rol pasivo por uno activo en el mantenimiento y mejora de su salud.

Garantizar la interoperatividad en un sistema distribuido es una de las necesidades fundamentales para facilitar el entendimiento entre las partes implicadas. La normalización en los distintos niveles de comunicación (sintáctico, semántico, organizativo, etc.) es la clave para la interoperatividad pero adoptar los esfuerzos normativos en ocasiones resulta una tarea ardua debido principalmente a la variedad de iniciativas normativas y el solapamiento entre ellas. Todos los aspectos relevantes de los sistemas distribuidos tanto de propósito general como específicos del dominio sanitario están cubiertos por esfuerzos normativos pero a menudo es necesario realizar una armonización entre iniciativas antes de aplicarlas al desarrollo de sistemas para no perder interoperatividad.

En esta Tesis Doctoral se investiga, diseña y desarrolla un paradigma de sistema distribuido orientado al sujeto de la asistencia que permite la colaboración de sistemas, usuarios, organizaciones y dispositivos con el objetivo común de mejorar y mantener la salud del sujeto de la asistencia concreto. Este paradigma hereda los fundamentos del concepto de organización virtual (Virtual Organization) y se ha denominado Person-Oriented Virtual Organization (POVO). El principal requisito de diseño de este paradigma es la adopción de normas y estándares que potencien la interoperatividad de los sistemas desplegados y garantice una larga vida útil de los mismos a través de la reutilización ulterior de sus capacidades. Siguiendo este principio de diseño la arquitectura de POVO está basada en las especificaciones del estándar ISO/EN 12967 (HISA) específico del dominio sanitario y el marco de trabajo RM-ODP. Para establecer una adecuada base arquitectural, en esta Tesis Doctoral se analiza la norma HISA y se reestructura para que sea más fiel a los principios de diseño del marco de trabajo RM-ODP y siga la formalización determinada por el estándar ISO 19793 (UML4ODP). La especificación del estándar HISA se extiende con capacidades de seguridad y de gestión de semántica. Dichas extensiones están

basadas en el análisis y armonización de la normativa aplicable buscando optimizar y facilitar la aplicación de la solución final.

Al margen de la especificación de los principios arquitecturales y funciones básicas de la POVO, se particulariza dicho paradigma para el estilo arquitectural SOA y la tecnología de computación en Grid y se diseña y desarrolla un mecanismo de control de acceso orientado a la administración por parte del sujeto de la asistencia y basado en capacidades semánticas. El mecanismo de autorización sigue un esquema de control de acceso basado en atributos que, utilizando ontologías y reglas de inferencia, permite automatizar el proceso de toma de decisiones. Así cualquier sujeto de la asistencia puede determinar de forma sencilla sus preferencias de acceso sobre los recursos e información directamente relacionados con su salud. Estas preferencias son traducidas e integradas en la base de conocimiento y un motor de inferencia será el que autorice o deniegue los intentos de acceso en base a las políticas definidas por el sujeto de la asistencia.

Las aportaciones de esta Tesis Doctoral, en líneas generales, ponen de manifiesto tres aspectos fundamentales en el ámbito de las TICs aplicadas al dominio sanitario. En primer lugar, el potencial que la normalización tiene para construir soluciones interoperables, reutilizables y con amplios horizontes temporales. Como consecuencia de ello es necesario potenciar las iniciativas normativas actuales y armonizar los solapamientos que existan entre ellas. Ejemplo de esto es la norma HISA cuya amplia adopción está ligada a la correcta integración con otras normas del mismo ámbito y su adecuada puesta en valor. En segundo lugar, los escenarios distribuidos con foco en el sujeto de la asistencia son el paso evolutivo natural de la asistencia sanitaria dadas las actuales (y futuras) coyunturas económicas y sociales. La tecnología está alcanzando una gran madurez en lo que a sistemas distribuidos se refiere pero aún queda camino por recorrer para poder construir soluciones fiables y eficientes que cubran los requisitos específicos de un escenario distribuido tan complejo como el que se presenta en esta Tesis Doctoral. Por último, el mecanismo de control de acceso diseñado y desarrollado sirve de prueba de concepto de cómo la tecnología actual puede otorgar a los individuos un papel activo en el mantenimiento de su salud y procesos relacionados sin necesidad de que tengan conocimientos tecnológicos avanzados.

Summary

The healthcare domain has been enormously influenced by the Information and Communication Technologies (ICTs) in last decades. Advancements on devices, information management, and care processes are consequences of that influence. In part due to technological heterogeneity of solutions and the lack of a formal methodology of ICT application, nowadays the healthcare scenario is fragmented into separated systems that rarely cooperate each other to provide advanced capabilities. This situation makes hard to achieve an improvement on the efficiency of processes, the evolution of the healthcare domain, and the decrease of costs. The clinical practice is also evolving to decentralized scenarios where the assistance of an individual is shared by different health organizations (often geographically distributed) and where coherency and privacy of information are essential requirements for an improvement of efficiency. In this decentralized scenario individuals should be the center of the processes, not organizations. Thus subjects of care will change their current passive roles for active ones in order to maintain and improve their health status.

Guarantee the interoperability of a distributed system is a crucial requirement to ease the understanding among the involved parties. The standardization at the different levels of communication (syntax, semantic, organization, etc.) is the key for the interoperability, but adopting standard efforts often is a hard task due to the spectrum of initiatives and the overlap between them. All the relevant features of distributed systems (of general purpose as well as specific of the health domain) are covered by normative efforts. However a harmonization among initiatives is needed before of applying them to the development of systems.

In this work of PhD Thesis a paradigm of distributed system is analyzed, designed and developed. This paradigm is oriented to the subject of care and allows the collaboration among systems, users, organizations, and devices with the common objective of improving and maintaining the health status of the particular subject of care. This paradigm inherits the foundations of the concept Virtual Organization and it has been named Person-Oriented Virtual Organization (POVO). The main design requirement of POVO is the adoption of standards that enhance the interoperability of deployed systems and guarantee a long life of them by means of the reutilization of their capabilities. By following this principle, the architecture of POVO is based on the specifications of the standard ISO/EN 12967 (HISA) and the framework RM-ODP. In order to establish a proper architectural base, the standard HISA is analyzed in this work and restructured to be conformed to the framework RM-ODP and the formalization of the standard ISO 19793 (UML4ODP). The specification of the standard HISA is extended with capabilities of security and semantic management. Such extensions are based on analysis and harmonization of suitable standards trying to optimize and ease the application of the final solution.

The POVO paradigm is particularized for SOA and Grid computing and an access control mechanism based on semantic capabilities and oriented to the management by the own subject of care is designed and developed. The authorization mechanism adopts an attribute-based access control schema that by

using ontologies and inference rules allows automating the making-decision process. Thus any subject of care could determine his/her preferences of access over his/her resources and information directly related with his/her health. These preferences are translated and integrated in a knowledge base and an inference engine may authorize or deny access attempts according to the policies defined by the subject of care.

The contributions of this work of PhD Thesis state the importance of three main features of ICT applied to the health domain. Firstly, standardization has an enormous potential for building interoperable and reusable solutions. As a consequence, it is needed to support the current normative initiatives and harmonizing the overlaps among them. The standard HISA is an example because its wide adoption depends on the proper integration with other standards within the health and architectural domains. Secondly, distributed scenarios centered on the subject of care are the next step of healthcare assistance due to the current (and future) economic and social situation. Distributed system technology is maturity enough to cover this kind of scenarios, but there is a long road ahead to achieve reliable and efficient solutions that cover the specific requirements of a distributed scenario as complex as presented in this work. Lastly, the access control mechanism designed and developed is a proof of concept of how current technology could empower citizens with an active role in the maintenance of their health with no necessity of technological advanced skills.

Glosario de abreviaturas

Abreviatura	Significado
AA	*Attribute Authority*
ABAC	*Attribute-based Access Control*
ACS	*Access Control System*
ADM	*Architecture Development Method*
ADO	*Access Decision Object*
ARIS	*Architecture of Integrated Information Systems*
BPM	*Business Process Management*
CEN	*European Committee for Standardisation*
CHF	*Connected Health Framework*
CIM	*Computation Independent Model*
CORBA	*Common Object Request Broker Architecture*
DA	*Descripción Arquitectural*
DAML	*DARPA Agent Markup Language*
DCOM	*Distributed Object Component Model*
DoDAF	*Department of Defense Architecture Framework*
DSL	*Domain Specific Language*
EBS	*Enterprise Service Bus*
Event-OA	*Event Oriented Architecture*
FEAF	*Federal Enterprise Architecture Framework*
FIPA	*Foundation for Intelligent Physical Agents*
GCM	*Generic Component Model*
HDTF	*Healthcare Domain Task Force*
HISA	*Health Information Service Architecture*
HIS-DF	*Health Information System – Development Framework*
HITSP	*Healthcare Information Technology Standards Panel*
HL7	*Health Level Seven*
HSSP	*Healthcare Services Specification Project*
IDL	*Interface Description Language*
IEC	*International Electrotechnical Commission*
IETF	*Internet Engineering Task Force*
IHE	*Integrating the Healthcare Enterprise*
ISO	*International Organization for Standardization*
ITU	*International Telecommunication Union*

LDAP	*Lightweight Directory Access Protocol*
M3A	*Me-As-An-Admin*
MDA	*Model Driven Architecture*
MDE	*Model Driven Engineering*
MoDAF	*Ministry of Defense Architectural Framework*
MOF	*Meta Object Facility*
OASIS	*Organization for the Advancement of Structured Information Standards*
OCL	*Object Constraint Language*
OGF	*Open Grid Forum*
OGSA	*Open Grid Services Architecture*
OMA	*Object Management Architecture*
OMG	*Object Management Group*
ORB	*Object Request Broker*
OSI	*Open Systems Interconnection*
OWL	*Web Ontology Language*
PAP	*Policy Administration Point*
PASS	*Privacy, Access and Security Services*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Point*
PHR	*Personal Health Record*
PIM	*Platform Independent Model*
PIP	*Policy Information Point*
POVO	*Person Oriented Virtual Organization*
PSM	*Platform Specific Model*
QoS	*Quality of Service*
QVT	*Query, Views and Transformations*
RAD	*Resource Access Decision Facility*
RBAC	*Role-Based Access Control*
RDF	*Resource Description Framework*
RFC	*Request For Comments*
RMI	*Remote Method Invocation*
RM-ODP	*Reference Model – Open Distributed Processing*
RPC	*Remote Procedure Call*
RUP	*Rational Unified Process*
SAML	*Security Assertion Markup Language*
SdA	*Sujeto de la Asistencia*

SLA	*Service Level Agreement*
SOA	*Service Oriented Architecture*
SOC	*Service Oriented Computing*
SOMF	*Service Oriented Modeling Framework*
SPARQL	*SPARQL Protocol and RDF Query Language*
STS	*Security token issuing and verification*
SWRL	*Semantic Web Rule Language*
SysML	*Systems Modeling Language*
TIC	*Tecnologías de la Información y Comunicación*
TOGAF	*The Open Group Architecture Framework*
UDDI	*Universal Description, Discovery, and Integration*
UML	*Unified Modeling Language*
UML4ODP	*Use of UML for ODP system specifications*
VO	*Virtual Organization*
WSDL	*Web Service Description Language*
WSRF	*Web Service Resource Framework*
XACML	*eXtensible Access Control Markup Language*
XDR	*External Data Representation*
XSD	*XML Schema Definition Language*
XSPA	*Cross-Enterprise Security and Privacy Authorization*

Sección 1. Introducción

Capítulo 1. Hipótesis y Objetivos

1. Introducción

1.1. La aplicación de las TICs en el dominio sanitario

El dominio sanitario se ha visto influenciado sobremanera por las posibilidades que ofrecen las Tecnologías de la Información y Comunicación (TICs) y que van desde la gestión electrónica y ubicua de datos sanitarios hasta el despliegue de servicios de generación de conocimiento. Estas capacidades avanzadas posibilitan un nuevo paradigma de asistencia sanitaria en el cual los procesos dejan de estar centrados en la organización sanitaria y pasan a tener al Sujeto de la Asistencia (SdA) como foco. Un ejemplo de esta transformación es la apuesta por desplazar, en la medida de lo posible, los tratamientos desde el hospital al hogar de los pacientes sobre todo en casos crónicos.

El cambio de modelo asistencial requiere de una mejor coordinación de los servicios sanitarios y sociales, reutilizando recursos, eliminando sistemas redundantes y disminuyendo los costes. La gestión de la información, por su parte, demanda que ésta pueda recuperarse allá donde esté y combinarse de manera coherente para apoyar los procesos de toma de decisiones. Por último, la aplicación y uso adecuado de las TICs facilita la migración del modelo tradicional de la asistencia en episodios con problemas de salud puntuales hacia la promoción de la salud y prevención de enfermedades, la auto-asistencia, los tratamientos crónicos y paliativos, etc. Resumiendo, la gestión eficiente de los sistemas de información sanitarios es crucial para alcanzar nuevos escenarios de asistencia sanitaria, ya sea considerando éstos como el elemento diferenciador que permite modificar la práctica clínica a través de la gestión ubicua y coherente de la información y la generación de nuevo conocimiento, o como componentes que soportan la asistencia y monitorización remotas.

1.2. Retos tecnológicos en el dominio sanitario

Aunque el potencial que tiene la aplicación de las TICs en salud es innegable, existen numerosos retos que deben ser abordados para lograr un uso eficiente de los sistemas de información sanitarios. Entre estos destacan la necesidad de mantener la coherencia entre datos duplicados, la reutilización de componentes o la convivencia de tecnologías heterogéneas. Estos y otros retos son consecuencia de la evolución independiente y fragmentada de los sistemas como resultado de coyunturas administrativas, económicas o legales [1]. Así, por ejemplo, la responsabilidad de los datos sanitarios a menudo está dividida entre diferentes instituciones y la coordinación de éstas puede ser difícil debido a restricciones administrativas o financieras. A esto se une la presión intensa para la rápida disponibilidad de los datos que a menudo contribuye al establecimiento de sistemas de información específicos a un servicio o especialidad. El resultado es la existencia de un gran número de aplicaciones y sistemas (bases de datos, dispositivos, etc.) aislados e incompatibles que están desplegados en las organizaciones sanitarias soportando necesidades específicas. Incluso dentro de una misma organización los sistemas de información están a menudo fragmentados en un conjunto de aplicaciones, datos y funcionalidades aisladas e inconsistentes entre ellas [2]. Disponer de un modelo integrado y coherente de sistemas cooperantes pasa por armonizar los distintos niveles de heterogeneidad (administrativa, de negocio,

tecnológica…) y alcanzar una interoperatividad completa que permita obtener las bondades que pueden derivarse del óptimo uso de las TICs.

La interoperatividad en el contexto sanitario necesita ser entendida en un sentido más amplio que el exclusivamente técnico, esto es, sirviendo al propósito de proporcionar una mejor, más segura y eficiente práctica sanitaria. La complejidad operativa del dominio sanitario es resultado de las siguientes características:

- Habitualmente la información necesaria para solucionar un problema está distribuida en distintos lugares (por ejemplo, en ambulatorios y hospitales).

- Los procesos asistenciales requieren la ejecución de tareas por parte de individuos con diversas habilidades y competencias, usualmente sin la supervisión de un coordinador único centralizado (profesionales de distintas especialidades, técnicos de maquinaria diagnóstica, cuidadores, etc.).

- Los problemas del ámbito médico son habitualmente bastante complejos con largos períodos de desarrollo (agudizándose para el caso de la población con condiciones crónicas).

- Se intercambia información y conocimiento sobre SdAs u otros temas sanitarios. Es de importancia capital entender esta información y actuar de forma coherente porque generalmente la salud de algún individuo es el tema central. Por ello es importante que los sistemas sean capaces de intercambiar la información (para poder recoger los datos de un sujeto en cualquier momento y desde cualquier lugar que se necesiten) pero más importante aún es que los sistemas entiendan y sepan procesar esa información de forma coherente ya que unos datos sin significado semántico (es decir, sin interpretación) son completamente inútiles.

- La comunicación de información sanitaria se puede dar entre actores (profesionales, SdAs, otros) y organizaciones que están separados cultural y lingüísticamente. Dos sistemas finales adaptados a las capacidades de sus respectivos usuarios deben ser capaces de compartir el mismo contexto semántico aun cuando el mismo término tenga representaciones diferentes (por ejemplo porque se utiliza en distintos lenguajes).

- Incluso dentro de una región donde no existen diferencias lingüísticas a menudo la comunicación se realiza a través de fronteras entre dominios administrativos. Dos sistemas sanitarios separados pueden haber resuelto terminologías diferentes para los mismos conceptos y la compartición de la semántica utilizada en ambos sistemas resulta imprescindible para poder establecer colaboraciones.

Este escenario heterogéneo en el que participan multitud de organizaciones se corresponde con el concepto tecnológico de sistema distribuido, en el cual un conjunto de elementos heterogéneos, distribuidos geográficamente e interoperables cooperan entre sí para ejecutar tareas complejas. Este

paradigma tecnológico presenta numerosas complejidades debido principalmente a los aspectos de interoperatividad y distribución y en los últimos años diversas iniciativas han desarrollado propuestas buscando facilitar el desarrollo y despliegue de tales sistemas. Se pueden encontrar así paradigmas arquitecturales como los agentes inteligentes, objetos distribuidos, las arquitecturas orientadas a servicios, computación en Grid, etc.; y tecnologías de implementación como los Servicios Web, RPC o CORBA. Aunque el uso de estos paradigmas y tecnologías facilita el desarrollo de un sistema distribuido, tareas como su diseño y construcción no son sencillas. Una práctica recomendada es utilizar metodologías estandarizadas que guíen en el proceso de desarrollo o marcos de trabajo que establezcan los parámetros y consideraciones de diseño necesarios en la construcción de sistemas distribuidos. Adicionalmente los sistemas distribuidos, debido a sus características intrínsecas, presentan un conjunto de riesgos potenciales a la seguridad como acceso no autorizado a componentes, suplantación de identidad, corrupción de los datos, violación de la privacidad en las comunicaciones entre componentes, etc. Es por ello crucial que los mecanismos de seguridad sean tenidos en cuenta desde la primera fase del diseño atendiendo a los riesgos potenciales. En el dominio sanitario donde la información a manejar e intercambiar es altamente sensible, implicando principalmente datos sanitarios de pacientes, las medidas de seguridad (y en concreto la gestión del control de acceso) deben ser cumplidas de manera estricta.

2. Estructura de la memoria

El presente documento se compone de las siguientes secciones y capítulos.

Sección 1. Introducción

- **Capítulo 1. Hipótesis y Objetivos:** presenta tras una breve introducción las hipótesis que motivan esta Tesis Doctoral y los objetivos planteados en la misma.

Sección 2. Estado de la Técnica

- **Capítulo 2. Paradigmas y tecnologías de sistemas distribuidos:** este capítulo presenta un conjunto de estilos arquitecturales y tecnologías de implementación para el diseño y despliegue de sistemas de computación distribuida. En primer lugar se describen las particularidades, beneficios y potencialidades de este tipo de sistemas para posteriormente describir estilos como las arquitecturas orientadas a servicios (SOA), los objetos distribuidos y los agentes. Se describe a continuación un conjunto de marcos arquitecturales para el diseño y desarrollo de sistemas distribuidos y abiertos de propósito general (RM-ODP, MDA, TOGAF y otros) y se plantean vías de armonización entre ellos. Dentro del ámbito sanitario se presentan varios marcos arquitecturales como HISA, HDF y HIS-DF así como algunas arquitecturas intermediarias (DHE, HDTF y HSSP). Finalmente, en cuanto a las tecnologías, este trabajo estudia cómo pueden contribuir a la implementación de sistemas de computación distribuida los Servicios Web, los mecanismos de gestión de semántica y la computación en Grid. Este capítulo termina con una comparativa entre

los métodos y estándares de formalización de arquitecturas cuyos resultados permiten identificar los puntos fuertes y débiles de las distintas iniciativas.

- **Capítulo 3. Métodos y técnicas de desarrollo:** existen diversas metodologías de desarrollo de sistemas que han alcanzado gran reconocimiento y que recogen las mejores prácticas en lo que a desarrollo de sistemas de propósito general se refiere. En este capítulo se revisan dichas metodologías y se realiza una armonización entre éstas y los principales marcos de trabajo presentados en el capítulo anterior.

- **Capítulo 4. El control de acceso y la gestión de privilegios:** este capítulo se centra en la seguridad de los sistemas de información distribuidos haciendo especial hincapié en el control de acceso y la gestión de privilegios. Se examinan y presentan las iniciativas más relevantes en cuanto a normalización en escenarios de propósito general así como en el dominio sanitario.

Sección 3. Resultados

- **Capítulo 5. Resultados:** este capítulo presenta las contribuciones de esta Tesis Doctoral. En primer lugar, elegida la norma HISA como base de la arquitectura de referencia, se presentan las conclusiones derivadas de su estudio así como las modificaciones a la misma para mejorar su conformidad con el modelo de referencia ODP. Paralelamente se define un marco de armonización entre los estándares e iniciativas relacionados con la seguridad. La formalización de la arquitectura basada en HISA se extiende con dicho marco normalizado además de con capacidades de gestión de semántica. Se diseña el paradigma de sistemas sanitarios distribuidos centrados en el SdA y con éste como el administrador del acceso a los recursos. Dicho paradigma se ha denominado POVO y utiliza como soporte la arquitectura normalizada basada en HISA con las extensiones de seguridad y gestión de semántica. Por último, se desarrolla en profundidad el mecanismo de control de acceso del escenario POVO para capacitar al SdA con las funciones de administrador del acceso a sus propios recursos.

Sección 4. Conclusiones y Anexos

- **Capítulo 6. Conclusiones y Línea de investigación:** en este capítulo se presentan las conclusiones que se han obtenido tras el desarrollo de esta Tesis Doctoral. Además describe la línea de investigación que el autor de esta Tesis seguirá a partir del momento de la obtención del título de Doctor.

- **Capítulo 7. Bibliografía:** recoge de forma ordenada las referencias utilizadas a lo largo de los capítulos anteriores.

- **Anexo. Aportaciones científicas:** el desarrollo de este trabajo ha dado lugar a la publicación de diversos artículos y capítulos de libros así como a la realización de aportaciones a diversos congresos de índole nacional e internacional, y en este capítulo se recogen los datos de las mismas.

3. Hipótesis

Dadas las características y necesidades concretas del ámbito sanitario y la aplicación eficiente de las TICs, las hipótesis en las que se basa esta Tesis Doctoral son las siguientes:

- Los escenarios centrados en el SdA poseen un enorme potencial para revolucionar la práctica asistencial haciéndola más eficiente y avanzada así como reduciendo los costes asociados. Los sistemas distribuidos son el paradigma de diseño que mejor se ajusta a estos nuevos escenarios.

- Ante la heterogeneidad de los sistemas y tecnologías existentes en la actualidad la normalización es la clave para potenciar la interoperatividad de las soluciones y resolver el problema de los sistemas fragmentados facilitando la construcción de sistemas avanzados que reutilizan las capacidades de otros.

- El rol activo de los SdAs en su salud es un aspecto clave de los nuevos escenarios de asistencia. La tecnología permite capacitar al SdA para que se implique en los procesos relacionados con su salud en diversos ámbitos. Uno de éstos es el de administración de los permisos de acceso a información sanitaria y personal, es decir, autorización y control de acceso.

4. Objetivos

El objetivo principal de esta Tesis Doctoral es contribuir a la formalización de arquitecturas de sistemas distribuidos en el dominio sanitario siguiendo el planteamiento de un nuevo paradigma de asistencia sanitaria centrado en el ciudadano que permita otorgarle mayor autonomía y un rol activo en la gestión de sus recursos sanitarios. Este objetivo general se particulariza en los siguientes objetivos concretos:

1. Armonizar los mecanismos, estándares y paradigmas relacionados con la seguridad (más concretamente con la autorización y el control de acceso) y aplicar el esquema armonizado resultante a un sistema de componentes distribuidos en el dominio sanitario.

2. Reestructurar la arquitectura de la norma ISO12967 para mejorar su conformidad con el modelo de referencia ODP y así optimizar su aplicabilidad como estándar de desarrollo de arquitecturas de sistemas distribuidos en el dominio sanitario y facilitar su aplicación.

3. Extender la arquitectura normativa anterior con aspectos de seguridad (en particular, control de acceso y gestión de privilegios) y con capacidades de gestión de semántica (ontologías, motores de inferencia, etc.), lo que facilitará la integración de estos servicios transversales con el resto de la organización.

4. Diseñar y desarrollar un paradigma que soporte la autonomía del SdA con éste como administrador de sus recursos y con capacidad para otorgar privilegios de acceso. Presentarlo como una prueba de concepto de la aplicación de tecnologías semánticas al control de acceso en el dominio sanitario y del potencial de las soluciones distribuidas donde el ciudadano toma un rol activo en sus procesos.

Sección 2. Estado de la Técnica

Capítulo 2. Paradigmas y tecnologías de sistemas distribuidos

1. Introducción a los sistemas distribuidos

Desde su aparición a mediados del siglo XX los sistemas computacionales han supuesto una revolución en numerosas disciplinas pese a que en su origen las computadoras tenían tales dimensiones y costes que su expansión estaba limitada a grandes empresas y operaban de forma independiente al no existir un medio de conexión entre ellas. Dos importantes avances en tecnología y comunicaciones cambiaron este escenario en la década de los 80. Por un lado comenzaron a desarrollarse microprocesadores integrados que llevaron el potencial de las grandes computadoras a ordenadores de pequeñas dimensiones y abarataron progresivamente los costes de fabricación. Por otro, surgieron las redes de ordenadores de alta velocidad (redes de área local o LAN) que permitían la interconexión de cientos de equipos en una misma localización y el intercambio de información entre ellos en un tiempo reducido. Inmediatamente les siguieron las redes de área extensa o WAN que elevaban exponencialmente el número de equipos interconectados y facilitaban la compartición de información entre cualquier par de puntos del globo terrestre. El resultado de estas tecnologías fue la posibilidad de construir redes de escala mundial como Internet y permitir trasvases de información y servicios como nunca antes había sido posible. Las redes como soporte tecnológico y la creciente capacidad de procesamiento de los equipos permitieron el despliegue de sistemas en los que se ejecutaban aplicaciones distribuidas entre equipos que cooperaban aun separados geográfica y administrativamente.

En la literatura se pueden encontrar numerosas definiciones del concepto “sistema distribuido” y una de las que mejor abarca la complejidad del término es la que lo define como aquel sistema en el que se pueden ejecutar tareas en distintas máquinas, pudiendo interactuar entre sí y facilitando de este modo el desarrollo de aplicaciones con procesos ejecutándose en una o varias máquinas remotas independientes y que ocultan esta circunstancia (en mayor o menor medida) a programadores y usuarios. Esta definición recoge dos de las principales características a las que aspira todo sistema distribuido:

1. La autonomía de sistemas: los componentes que forman parte del sistema distribuido colaboran entre sí para proporcionar funcionalidades complejas y sofisticadas pero manteniendo siempre un alto grado de independencia lo que permite obtener un sistema flexible y adaptable ante cambios evolutivos de los distintos componentes. Así un sistema puede estar distribuido entre varias autoridades y organizaciones autónomas de gestión sin que exista un único punto de control. Además la autonomía de los sistemas puede facilitar que el comportamiento erróneo de un componente quede aislado y no se propague por el sistema haciendo que fallen el resto de elementos.

2. La transparencia de distribución al usuario final: permite ocultar la complejidad inherente a la comunicación y operación del sistema distribuido presentando al usuario final un comportamiento similar al de un sistema centralizado de enormes capacidades y localizado en el mismo lugar que él. De esta forma el usuario no es consciente, por ejemplo, de cuán remotos están los recursos a los

que accede o el número de transformaciones de información que se llevan a cabo para solventar las fronteras de heterogeneidad tecnológica y/o administrativa.

Otras características de los sistemas distribuidos son: ubicación en lugares distantes, los componentes de un sistema distribuido pueden estar ubicados en distintos puntos en el espacio y las interacciones entre componentes podrán ser locales o remotas; concurrencia, todo componente de un sistema distribuido puede ejecutar acciones en paralelo con cualquier otro componente; asincronía, las actividades de comunicación y procesamiento no son regidas por un reloj global único; heterogeneidad, no está garantizado que los componentes de un sistema distribuido estén construidos sobre la misma base tecnológica y el conjunto de las diversas tecnologías seguramente cambiará con el transcurso del tiempo; y movilidad, las fuentes de información, los nodos de procesamiento y los usuarios pueden ser físicamente móviles. Los programas y los datos pueden también ser desplazados entre los nodos, por ejemplo para estar en consonancia con la movilidad física o para optimizar la calidad de funcionamiento.

Todas las características inherentes a los sistemas distribuidos revierten directa o indirectamente en requisitos de interoperatividad, entendiendo ésta como la habilidad de dos entidades para colaborar entre sí. Dicho de otro modo, la capacidad efectiva de comunicar información mutuamente con el objetivo de intercambiar proposiciones, peticiones, resultados y acuerdos. El término puede ser referido a distintos niveles de complejidad y/o aspectos funcionales de la comunicación pero la definición de los mismos no es un punto consensuado en la literatura. Una clasificación bastante extendida de niveles de interoperatividad puede verse en la Figura 2.1 [3]. El nivel más elemental de interoperatividad es el técnico, el cual centra la conectividad entre servicios computacionales permitiendo que la información sea transportada entre máquinas. Sobre los datos intercambiados se deben especificar el nivel sintáctico (el lenguaje común en su formato) y semántico (que permitirá extraer el significado del conjunto de los datos y que sea igualmente entendible por el remitente y receptor). Un nivel más arriba encontramos la interoperatividad pragmática especificando el deseo o voluntad de las partes de realizar las acciones necesarias para conseguir una colaboración efectiva. Finalmente el nivel organizacional especifica la habilidad de las organizaciones a las que pertenecen los sistemas de comunicar y transferir de forma efectiva datos o información aun cuando pueda potencialmente usar una amplia variedad de sistemas de información diferentes sobre infraestructuras dispares. Como la heterogeneidad de sistemas puede darse en cada uno de estos niveles, el objetivo es ofrecer una interoperatividad organizacional la cual aglutina y se apoya en el resto de niveles.

Para manejar de forma eficiente la complejidad de la distribución de sistemas es fundamental organizarlos adecuadamente y formalizar lo que se denomina la arquitectura del sistema. Dicha arquitectura define, por un lado, la organización lógica y las interacciones de los componentes del sistema y, por otro, la organización física para implementar dicha arquitectura sobre máquinas y redes reales. Para facilitar la implementación los sistemas se suelen organizar en base a una capa software que se sitúa lógicamente entre las aplicaciones de alto nivel y las capas más bajas que ofrecen las

facilidades básicas de comunicación y sistema operativo. Esta capa se suele conocer como software de intermediación (o *middleware*) y ofrece al programador un conjunto de soluciones estándar escogidas y basadas en la práctica. De este modo el diseñador de una aplicación puede trabajar en un mundo que es transparente a la complejidad de la distribución del sistema. La Figura 2.2 muestra el esquema de un sistema distribuido que consta de tres equipos cada uno con sus propias características hardware y software que pueden ser diferentes a las del resto de componentes del sistema. Es la capa de distribución la que independiza las aplicaciones de las configuraciones reales de los equipos y permite que éstos cooperen y se comuniquen entre sí.

Figura 2.1. Escala de niveles de interoperatividad según [3]

La transparencia de distribución se divide, a su vez, en un conjunto de transparencias de modo que cada una oculta al programador o usuario aspectos concretos del funcionamiento y configuración de los sistemas distribuidos.

- Transparencia de ubicación: enmascara la utilización de información sobre la ubicación espacial de los componentes del sistema. Esta transparencia de distribución proporciona una identificación lógica de los componentes independiente de la ubicación física real de los mismos.

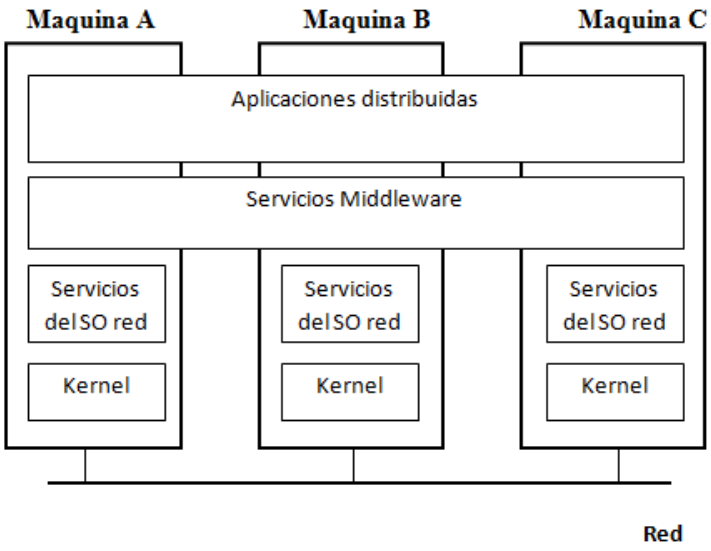

Figura 2.2. Ejemplo de sistema distribuido

- Transparencia de fallos: oculta con respecto a un componente del sistema el fallo y la posible recuperación de otros (o de él mismo) para permitir la tolerancia a los malos funcionamientos. Cuando se proporciona esta transparencia de distribución el diseñador puede trabajar en un mundo ideal en el que los fallos no se propagan.

- Transparencia de acceso: enmascara las diferencias en la representación de datos y mecanismos de invocación para permitir la comunicación entre componentes. Esta transparencia de distribución resuelve muchos de los problemas de interoperatividad entre sistemas heterogéneos y además permite acceder a recursos locales y remotos utilizando las mismas operaciones.

- Transparencia de movilidad: enmascara, con respecto a un componente, la posibilidad de que se cambie la ubicación del mismo sin afectar a su funcionamiento.

- Transparencia de rendimiento: permite que a medida que la carga varía el sistema se reconfigure para mejorar el rendimiento. Se utiliza a menudo para obtener equilibrio de carga y reducir la latencia.

- Transparencia de reubicación: permite que el sistema continúe su operación aun cuando algunos componentes sean sustituidos o migren mientras están en uso.

- Transparencia de replicación: enmascara la utilización de un grupo de componentes, mutuamente compatibles en comportamiento, para soportar una misma interfaz. La replicación suele utilizarse para mejorar la calidad de funcionamiento y la disponibilidad.

- Transparencia de persistencia: permite ocultar la desactivación y reactivación de componentes almacenando el estado y los datos previos a la desactivación para una vez reactivado el componente continuar su proceso como si no hubiera dejado de funcionar.

- Transparencia de transacción: enmascara la coordinación de actividades dentro de una configuración de componentes para asegurar la consistencia. Esta puede incluir la transparencia de concurrencia que oculta a un componente que el uso de un recurso se está haciendo de modo compartido con otros componentes que compiten por él y asegura que el recurso se encuentra en un estado consistente.

- Transparencia de escalabilidad: permite al sistema y las aplicaciones cambiar de escala (dar servicio a mayor número de usuarios) sin cambios en la estructura del sistema o los algoritmos de las aplicaciones.

Dados los beneficios que aportan los sistemas distribuidos se han realizado numerosos esfuerzos por desarrollar tecnologías y métodos que permitan de forma efectiva desplegar este tipo de sistemas con las transparencias comentadas. Fruto de estas iniciativas se cuenta actualmente con paradigmas de diseño de sistemas que facilitan su implementación en base a las mejores prácticas en este ámbito (objetos distribuidos, agentes inteligentes, arquitecturas orientadas a servicios…) y con tecnologías que soportan las transparencias de distribución presentadas y simplifican el despliegue de sistemas distribuidos (como CORBA, los Servicios Web o los Servicios Grid). Además existen numerosos marcos de representación formal (como RM-ODP, TOGAF o MDA) que buscan simplificar las tareas de diseño, desarrollo, despliegue, mantenimiento y evolución de sistemas distribuidos, aportando además los principios de apertura e interoperatividad a través de estandarización y lenguajes de descripción formal.

El resto del presente capítulo aborda los sistemas distribuidos desde varias perspectivas. En primer lugar se describen algunos de los paradigmas más relevantes actualmente para el diseño de sistemas distribuidos como son las orientaciones a objetos y a servicios, los modelos de agentes y la orientación a eventos o mensajes. El segundo punto que se aborda es el de los métodos de diseño de arquitecturas y su especificación formal como RM-ODP, MDA o TOGAF. Además se incluyen algunos de los lenguajes de formalización más utilizados como UML o SysML. Posteriormente se describen las principales tecnologías específicas para la distribución de componentes como CORBA y las tecnologías de Servicios Web. Finalmente atendiendo a que el dominio de desarrollo del presente trabajo de Tesis Doctoral es el sanitario se presenta un conjunto de herramientas de formalización de arquitecturas y software de intermediación aplicados a este ámbito.

1.1. Los sistemas distribuidos en el ámbito sanitario

Actualmente los términos "*ICT for Health*", "*Health Informatics*" y "*eHealth*" cubren todo el espectro de aplicaciones TICs con influencia en el sector sanitario y van más allá de la exclusiva asistencia a los pacientes. Una apropiada aplicación de las TICs en salud tiene el potencial de proporcionar a los

ciudadanos un acceso mejorado a sistemas sanitarios más eficientes y sofisticados así como implicarlos en el cuidado y mantenimiento de su propia salud, haciendo que tomen un rol activo en su bienestar. Las TICs mejoran la calidad y eficiencia de los servicios sanitarios y aportan un conjunto de beneficios económicos ya que permiten, por ejemplo, reducir los costes del tratamiento de los pacientes en hospitales así como derivarlos y monitorizarlos en sus propios hogares (por ejemplo en condiciones crónicas). La adopción de los servicios de *eHealth* pasa por la aceptación y confianza en ellos lo cual depende del correcto funcionamiento de los servicios y aplicaciones así como de la correcta protección de información y recursos sensibles.

A todo esto se une que el entorno de procesamiento de información dentro de las organizaciones sanitarias se está haciendo cada vez más complejo en parte debido a la heterogeneidad de las tecnologías empleadas y las aplicaciones de usuario altamente interactivas. Un prerrequisito asumido para alcanzar la interoperatividad completa es la estandarización basada en el consenso entre todas las partes implicadas. La normalización permitirá diseñar y construir sistemas abiertos y completamente interoperables, facilitando además la integración de sistemas que previamente estuvieran desplegados en la organización sanitaria. Existen numerosos estándares e iniciativas para la formalización de arquitecturas de sistemas distribuidos de propósito general los cuales, en su mayoría, han sido desarrollados de forma independiente y siguen líneas de trabajo separadas entre sí. Debido a que el ámbito de la salud posee características distintivas, será interesante trabajar con aquellos marcos arquitecturales y metodologías de formalización específicos del entorno sanitario. En este sentido utilizaremos como punto de partida el estándar ISO-12967 (HISA) [2] que surgió como norma europea y ha sido recientemente adoptado por la ISO (*International Organization for Standardization*) [4] y que está basada en el marco de trabajo para el desarrollo de estándares de sistemas distribuidos RM-ODP [5].

2. Paradigmas de computación distribuida

2.1. Orientación a objetos e ingeniería de sistemas basada en componentes

En la década de los 90 se comienza a utilizar el paradigma de orientación a objetos para el desarrollo de soluciones de computación distribuida. Este paradigma poseía un conjunto propio de principios que dividen la lógica de solución de los sistemas en objetos separados y describen un tipo específico de relación entre ellos que les lleva a colaborar y comportarse como una solución completa. Cada objeto contiene datos (atributos) y procedimientos (métodos) encapsulados para representar una entidad, y para cada uno existe una interfaz de objeto a través de la que se puede interactuar con el mismo. Los conceptos y principios básicos del diseño orientado a objetos son [6]:

- Objeto/clase: representación de una entidad en el mundo real, la cual contiene información y ofrece servicios. Un sistema se compone de objetos interactuantes cada cual sirviendo a una

función separada y definido por sus propiedades. Las clases recogen las lógicas de solución y los objetos son la representación de las clases en tiempo de ejecución.

- Separación entre objetos, es un principio regido por dos propiedades de los objetos: encapsulación, por la cual la información contenida en un objeto es accesible solamente mediante interacciones en las interfaces soportadas por el objeto; y abstracción, que implica que los detalles internos de un objeto están ocultos a otros objetos. Este principio resulta de capital importancia para el tratamiento de la heterogeneidad pues permite implementar servicios de maneras diferentes, utilizando cualquier mecanismo y tecnología, lo que hace posible la portabilidad y potencia la interoperatividad. Estas dos propiedades permiten además sustituir componentes de forma transparente al usuario u otros objetos con los que interactúan siempre que se mantengan las interfaces de interacción.

- Herencia: habilidad de una clase de extender o sustituir funcionalidades de otra clase. La también llamada subclase tiene una sección que es derivada (heredada) de la superclase además de su propio conjunto de funciones y datos. Una subclase puede hacer cualquier cosa que realiza la superclase y además puede ser extendida con funcionalidad única (a través de un proceso llamado especialización).

- Polimorfismo: capacidad de reemplazar un objeto con múltiples sub-objetos cada uno con una implementación diferente.

- Principio Abierto Cerrado (*Open-Closed Principle,* OCP): establece que las clases pueden ser extendidas y modificadas siempre que se mantengan las interfaces de operación, es decir, mantener interfaces abstractas de los métodos. Este es un requisito de diseño clave que ayuda a proteger la funcionalidad reusable de los objetos sobre la cual múltiples clientes pueden tener dependencias.

- "No te repitas" (*Don't repeat yourself*, DRY): este principio determina que, si existe lógica potencialmente reutilizable, debería estar separada de manera que se facilite su reutilización.

- Principio responsabilidad única (*Single Responsibility Principle*, SRP): los objetos están idealmente centrados en un único propósito y este principio apuesta por limitar el alcance funcional de los objetos para que su interfaz sólo tenga que ser modificada si ese único propósito que persiguen cambia.

- Delegación: si un objeto requiere de lógica que ya existe en otro objeto, entonces el primero debería dar (delegar) la responsabilidad de ejecutar esa lógica al otro objeto en lugar de hacerlo él mismo.

Paralelamente al diseño orientado a objetos surgió el paradigma de ingeniería del software basada en componentes [7]. Esta propuesta pretendía facilitar el desarrollo software en un momento en que la funcionalidad de los sistemas estaba creciendo exponencialmente. Para ello enfatiza la división de la funcionalidad de los sistemas por temas (por ejemplo, tema de seguridad, de interfaz de usuario, de acceso a recursos, etc.) y el diseño y desarrollo de cada uno por separado sin olvidar las relaciones entre ellos. Un componente individual es un paquete software o módulo que encapsula un conjunto de funciones (o datos) relacionadas. Todos los procesos del sistema son localizados en componentes separados de tal forma que todas las funciones y datos dentro de cada componente están relacionados semánticamente. Los componentes se comunican a través de interfaces que abstraen de la implementación interna de los componentes y sólo exponen lo que hace el componente y qué necesita para funcionar. Las interfaces se dividen en dos tipos: "interfaz proporcionada" (*provided*, la que publica un componente que ofrece servicios al resto del sistema) e "interfaz utilizada" (*used*, que especifica los servicios que un componente necesita). El diseño basado en componentes comparte varios principios con el orientado a objetos como la encapsulación, el uso de interfaces y la reutilización de componentes.

La filosofía de orientación a objetos se ha diversificado en los últimos años en un conjunto de paradigmas que se describen brevemente a continuación. Aunque siguen aproximaciones diferentes en sus características intrínsecas pueden encontrarse elementos ya descritos en la orientación a objetos. Se resumen tres de las propuestas más relevantes: orientación a servicios, paradigma de agentes y orientación a eventos o mensajes.

2.2. La computación y arquitecturas orientadas a servicios

El término de computación orientada a servicios (*Service Oriented Computing*, SOC) abarca un amplio espectro de elementos que van desde un paradigma concreto y principios de diseño a un modelo arquitectural único, conceptos, tecnologías y marcos de trabajo relacionados. La base de la SOC es el *servicio* el cual puede ser definido como una unidad de trabajo bien delimitada, encapsulada y realizada por un componente a través de una interfaz prescrita y consistente con unas restricciones y políticas especificadas en la descripción del servicio [8]. La intención es implementar cierta funcionalidad solamente una vez y hacerla ampliamente accesible. De esta forma, el servicio podrá ser directamente reutilizado en el futuro y además para el usuario del mismo los posibles cambios de implementación serán transparentes. Estos principios son heredados de la orientación a objetos y la única diferencia es el nivel de detalle en el que se trabaja siendo en general un objeto un elemento más simple que un servicio. Dicho esto, el paradigma de diseño de orientación a servicios tiene como objetivo crear unidades lógicas de funcionalidad, individualmente contenidas, que pueden ser utilizadas repetidamente y combinarse para soportar funcionalidades más complejas. La orientación a servicios como paradigma de diseño consiste en los siguientes principios:

1. Contrato de servicio estandarizado: en el cual se exponen el propósito y las capacidades del servicio así como pre y post condiciones de uso. Un contrato puede constar de un conjunto de documentos que describen el servicio, por ejemplo en los Servicios Web tenemos documentos *WSDL*, *OWL-S*, *WS Policy* y *SLA* (*Service Level Agreement*). El objetivo de este principio es proveer a los servicios con un nivel de interoperatividad natural dentro de las fronteras de un inventario de servicios. Además facilita que las capacidades y propósitos de los servicios sean fácil e intuitivamente entendidas.

2. Bajo acoplamiento: reducción de dependencias entre servicios, sus implementaciones o los clientes. Promueve un entorno en el cual los servicios y sus clientes pueden evolucionar y adaptarse en el tiempo con un mínimo impacto en el resto de elementos.

3. Abstracción de servicio: busca enmascarar tanto como se pueda los detalles subyacentes de un servicio facilitando el bajo acoplamiento. El objetivo es mantener el contrato de servicio conciso y prevenir el acceso a detalles del servicio que son innecesarios para su utilización. En la SOC el término *encapsulación* tiene distinto significado que el de *abstracción*, como en la orientación a objetos. El primero identifica la lógica, recursos e información que forman parte del servicio, así un servicio es un contrato y todo aquello que encapsula. Por otro lado, abstracción distingue las partes encapsuladas del servicio que son expuestas y las que se ocultan a los consumidores del servicio.

4. Reutilización de servicio: enfatiza la concepción de servicios como recursos de empresa con funcionalidad independiente de contexto y mide el potencial que presentan los servicios de ser reutilizados para otros propósitos. Permite que la lógica del servicio sea heredada repetidamente en el tiempo dando valor extra a la inversión. Además incrementa las capacidades de negocio permitiendo utilizar servicios antiguos a través de la composición y dando soporte a nuevos procesos de negocio.

5. Autonomía de servicio: para que los servicios presenten sus capacidades de forma consistente y fiable (además de predecibles en tiempo de ejecución), la lógica de solución subyacente a ellos necesita tener un grado de control significativo sobre su entorno y recursos.

6. Servicios sin estado: la gestión de excesiva información de estado puede comprometer la disponibilidad de un servicio y minar su potencial de escalabilidad. La “información de estado” se compone de datos asociados con una actividad actual en proceso y la “gestión del estado” representa el procesamiento de esa información. Este principio facilita la escalabilidad del sistema y soporta el diseño de lógica de servicio independiente de contexto además de potenciar la reutilización.

7. Publicación y descubrimiento de servicios: para que los servicios puedan ser eficientemente reutilizados y entendidos como recursos de la empresa, necesitan poder ser identificados y entendidos cuando se dé la ocasión de utilizarlos. La publicación y descubrimiento de servicios requieren de un mecanismo consistente para comunicar meta-información sobre los mismos (recursos que utilizan, su funcionalidad y modo de operación, etc.), que esta información esté definida con precisión para entender los servicios además de almacenada y centralizada en un formato consistente, y que se pueda acceder a los repositorios de meta-información y encontrar servicios en base a criterios de búsqueda. Esta meta-información sobre los servicios incluye el propósito de los mismos, sus capacidades y las limitaciones de éstas.

8. Composición de servicios: mientras la sofisticación de las soluciones orientadas a servicios crece, también lo hace la complejidad de las configuraciones de composiciones de servicios subyacentes. Aplicando este principio en un entorno se consigue que la funcionalidad de los servicios exista como unidades componibles, permitiendo reutilizar esas unidades y construir funcionalidades compuestas para resolver nuevos problemas.

Finalmente, una plataforma de computación orientada a servicios se basa en una arquitectura orientada a servicios (*Service Oriented Architecture,* SOA) que establece el modelo arquitectural basado en servicios. Una implementación SOA puede consistir en una combinación de tecnologías, productos, APIs, extensiones a la infraestructura de soporte y otros elementos. La implementación real de SOA es única para cada empresa pero siempre comparten elementos tecnológicos y de plataforma para la creación, ejecución y evolución de soluciones orientadas a servicios. Todo esto se resume en decir que la principal característica de SOA es que es agnóstica con respecto a la tecnología subyacente de implementación.

Los servicios pueden ser categorizados por el tipo de lógica que encapsulan, el potencial de reusabilidad que esta lógica tiene y cómo se relaciona con los dominios existentes en la empresa. Basándose en esto se tiene una clasificación de servicios en tres tipos:

- Servicios de entidad (*entity services, entity-centric business services* o *business entity services*): centrados en el negocio basan su funcionalidad en las entidades de negocio de la empresa y son altamente reutilizables porque son independientes de los procesos de negocio que los utilizan.

- Servicios de tarea (*task services*): servicios de negocio con una capacidad funcional directamente relacionada con una tarea de un proceso de negocio con lo que son potencialmente menos reutilizables.

- Servicios de utilidades (*utility services*): proporcionan funcionalidades reutilizables y transversales a todos los procesos de negocio como el registro de eventos o el manejo de excepciones.

Una SOA representa un estilo arquitectural para la construcción de sistemas basados en servicios. Por ello la funcionalidad de la arquitectura está repartida entre ellos, los cuales pueden ser invocados y

cuyas descripciones de interfaz pueden ser publicadas y descubiertas. La arquitectura tiende a un escenario en el que sus elementos no son más que proveedores o consumidores de servicios, los cuales pueden combinarse para obtener nuevos o mejorados procesos de negocio.

Los beneficios que justifican la popularidad del paradigma SOC son la promoción de la interoperatividad (la SOC otorga un conjunto de transparencias que reducen la complejidad de comunicación ente servicios), la facilidad de federación (recursos y aplicaciones que colaboran manteniendo su autonomía y autogobierno a través del despliegue de servicios estandarizados y componibles, cada uno encapsulando procedimientos concretos de la empresa y expresándolos de manera consistente), el mayor número de opciones gracias a la diversificación de proveedores, el incremento en la rentabilidad de la inversión mediante la reutilización de los servicios desarrollados, el incremento de la agilidad organizacional y la optimización del uso de TIC (aplicando SOC se reduce la redundancia, el tamaño y los costes operacionales de la plataforma tecnológica). Por otro lado, los obstáculos que introduce la orientación a servicios son la complejidad de diseño de la arquitectura y de los servicios (que tienen que ser reutilizables e independientes de contexto, propósito, etc.), la necesidad de utilizar estándares de diseño y la exigencia de congeniar políticas de gestión sobre servicios cuando son desarrollados por una entidad y reutilizados por otras con diferentes propósitos.

	Orientación a objetos	**Orientación a servicios**
Lógica de solución en diseño	Clases	Servicios
Lógica de solución en ejecución	Objetos	Instancias de servicios
Funciones	Métodos	Capacidades
Datos de estado	Atributos de métodos	No, los servicios son sin estado
Especificación de elementos	Interfaz	Contrato de servicio
Principios	Encapsulación	Abstracción de servicio
	Herencia	-
	Generalización / Especialización	Niveles de granularidad
	Abstracción de objeto	-
	Polimorfismo	-
	Principio Abierto-Cerrado (OCP)	Principio Abierto-Cerrado (OCP)
	No te repitas (DRY)	Modelos agnósticos de servicios
	Principio responsabilidad única (SRY)	Cohesión
	Delegación	Delegación
	Asociación	-
	Composición	-
	Agregación	-

Tabla 2.I. Fundamentos de las orientaciones a objetos y servicios

2.2.1. Orientación a objetos vs orientación a servicios

Si bien la orientación a servicios está inspirada en la orientación a objetos, la diferencia entre ambos paradigmas radica en los principios de diseño a aplicar, obteniendo por tanto diferentes tipos de características de diseño. Algunos principios del paradigma de orientación a objetos fueron eliminados y otros nuevos añadidos a la orientación a servicios. Por otro lado, algunos son heredados y trasladados a la orientación a servicios con un cambio de nomenclatura. En la Tabla 2.I se puede ver una correlación entre los principios de ambos paradigmas.

De esta comparativa es necesario destacar que uno de los fundamentos básicos de la SOC es la autonomía de los servicios que tiene como consecuencia la reducción de las relaciones entre ellos. Así no se utilizan principios de la orientación a objetos como el de herencia, polimorfismo, asociación, composición y agregación, que suponen un acoplamiento rígido entre elementos. En su lugar la SOC opta por composición de servicios en tiempo de ejecución a través del descubrimiento de los mismos. Un punto que merece ser aclarado de esta comparativa es que el principio de encapsulación en la orientación a objetos es la propiedad por la cual la información contenida en un objeto es accesible solamente mediante interacciones en las interfaces soportadas por el objeto. Es un principio que establece que un objeto sólo debería ser accedido a través de una interfaz pública y que su implementación debería quedar oculta para otros objetos. Esta noción de diseño es comparable con el principio de abstracción en la orientación a servicios, el cual también está centrado en la deliberada ocultación y aislamiento del funcionamiento interno del servicio con respecto a otros. Este principio, aunque de mismo nombre, no tiene que ver con la abstracción de objetos. El propósito de ésta es crear clases simplificadas que escondan la complejidad de la implementación y exponga sólo los métodos y atributos necesarios. Este principio de abstracción se utiliza para soportar la herencia para la definición de clases abstractas. Conceptualmente ambos principios son similares pero como el de orientación a servicios no soporta la herencia no hay una noción correspondiente de una clase abstracta.

2.3. Agentes y sistemas multi-agentes

Los agentes surgen como una evolución de los sistemas expertos de los 70 y los sistemas cooperantes (también conocidos como inteligencia artificial distribuida) de los 80 y 90. Un agente es un proceso computacional autónomo, con iniciativa y capacidad de explorar y modificar su entorno y posibilidad de comunicarse con otros agentes [9]. Los agentes están sujetos a varias clasificaciones según si son, por ejemplo, reactivos o cognitivos, si interaccionan con otros agentes, con el entorno o con personas, según la naturaleza del entorno en el que operan (especial o general), etc. Las arquitecturas de construcción de agentes se dividen en: deliberativas (existe planificación, permiten agentes proactivos), reactivas (sin modelo del entorno) o híbridas. La necesidad de desarrollar aplicaciones complejas compuestas de multitud de subsistemas que interaccionan entre sí obliga a distribuir la inteligencia entre agentes y a construir “sistemas multi-agente” que permiten la gestión inteligente de un sistema complejo coordinando los distintos subsistemas que lo componen e integrando los objetivos particulares de cada subsistema en un objetivo común. Cada subsistema sigue siendo autónomo y

mantiene capacidades de decisión locales pero a su vez existen unas políticas de cooperación, coordinación y negociación entre agentes que deben ser observadas.

Un sistema multi-agente cooperante está formado por un conjunto de agentes cada uno con sus propias capacidades y especializado en tareas determinadas. Existe una misión común que puede descomponerse en tareas independientes para paralelización y cada agente tiene un conocimiento limitado del entorno, de la misión del grupo y de las intenciones del resto de agentes. Pese a la mayor complejidad de desarrollo que supone un sistema multi-agente frente a otros sistemas centralizados, la distribución de las decisiones posee muchas ventajas:

- mayor autonomía para los agentes individuales y descentralización de la toma de decisiones,
- no toda la información debe alcanzar a un agente,
- es posible realizar acciones coordinadas,
- organización dinámica de la arquitectura lo que conlleva una mayor capacidad de adaptación del sistema multi-agente a cambios en el entorno, y
- diversidad en las relaciones entre agentes (conocidos, de comunicación, de subordinación, operativa, de información, de conflicto, competitiva, etc.).

La coordinación en los sistemas multi-agente se realiza a través de mecanismos de registro y descubrimiento utilizando lenguajes de descripción de servicios para exponer y consultar las capacidades aportadas por los distintos servicios. Como ejemplo podemos contar con el lenguaje WSDL [10] basado en XML [11] que separa los servicios definidos en términos abstractos de los formatos de datos y protocolos concretos utilizados para su implementación. Otro ejemplo de lenguaje de descripción de servicios que se suele utilizar en entornos multi-agente es el OWL-S [12], el cual hace uso de una ontología en OWL [13] para describirlos. Permite automatizar el descubrimiento, invocación, composición y monitorización de la ejecución.

2.3.1. Agentes y Arquitecturas orientadas a objetos

El paradigma de agentes puede verse como una especialización de la orientación a objetos ya que si bien poseen algunos fundamentos comunes también presentan algunas diferencias notables. Entre las similitudes podemos destacar que ambas soluciones abogan por la distribución de entidades que componen el sistema en un entorno abierto. Se presenta entonces un escenario con múltiples elementos (agentes u objetos) heterogéneos y de distinta naturaleza que habrá que descubrir y con los que se podrá interactuar a través de interfaces formalmente definidas. Además, ambos modelos trabajan con sistemas complejos que presentan un considerable número de comportamientos y gran cantidad de información que cambia con el tiempo.

Frente a estos puntos comunes encontramos como principal diferencia que en los sistemas multi-agente las distintas entidades pueden relacionarse por competición y sólo la intención de lo que se pretende es conocida, quedando indeterminada para cada agente la dinámica del sistema. En la orientación a objetos no se especifica el modo de operación pero se tiende a la cooperación de objetos e incluso a la composición de los mismos para ofrecer capacidades avanzadas.

2.4. Orientación a eventos o mensajes

Mientras que en las arquitecturas orientadas a objetos éstos eran los elementos principales, en las arquitecturas orientadas a eventos (Event-OA) [14] el objetivo es promover la producción, detección, consumo y reacción a eventos. Un *evento* puede ser definido como un cambio de estado significativo que merece atención por el sistema. Este patrón arquitectural puede aplicarse en el diseño e implementación de sistemas distribuidos que transmiten eventos entre servicios y componentes software con bajo acoplamiento entre sí. Un sistema distribuido orientado a eventos consiste típicamente en un conjunto de emisores de eventos y otro de consumidores. Éstos tienen la responsabilidad de reaccionar tan pronto como se presente el evento y esa reacción puede ser completamente proporcionada por el consumidor o derivada (filtrada, transformada y reenviada) a otros. Si los consumidores reaccionan de forma independiente a los eventos entonces se habla de arquitecturas orientadas a mensajes mientras que si los eventos motivan la comunicación y cooperación entre componentes entonces se amplía la denominación a arquitecturas orientadas a eventos.

Los sistemas y aplicaciones construidos mediante el patrón orientado a eventos permiten determinar más fielmente la respuesta de los mismos ante los eventos ya que estas arquitecturas, por diseño, están más normalizadas que los entornos orientados a mensajes los cuales son más imprevisibles. Por su naturaleza, las arquitecturas orientadas a eventos poseen un muy bajo acoplamiento entre componentes y son altamente distribuidas. La fuente de un evento sólo conoce el evento que transmite a la arquitectura, ignorando el proceso que se va a realizar de dicho evento, las partes o consumidores interesados en el mismo o las reacciones que se sucederán. El rastreo de un evento a través de una red puede ser, por tanto, complicado. Por ello las arquitecturas orientadas a eventos encajan mejor con los flujos de información y trabajo asíncronos.

Las arquitecturas orientadas a eventos pueden complementar a aquellas orientadas a servicios porque éstos pueden ser activados mediante disparadores asociados a los eventos que se sucedan. Esta combinación de patrones se denomina arquitectura orientada a servicios y conducida por eventos (*Event-driven SOA*).

3. Métodos de diseño de arquitecturas y especificación formal

3.1. Introducción

Los sistemas distribuidos heterogéneos de gran escala son, por sus propias características, mucho más complejos de diseñar, desarrollar, mantener y documentar que los clásicos sistemas centralizados y homogéneos. A comienzos de la década de los 90 comienzan a surgir iniciativas que buscan reducir esa complejidad [15][16]. Dichas aproximaciones siguen distintos fundamentos como la descomposición del sistema en componentes más simples y manejables, el uso de modelos que simplifican los sistemas reales, o la focalización del diseño en áreas de interés (o puntos de vista) cada una centrada en un aspecto concreto del sistema (interacciones entre componentes, tecnología, procesos, etc.). El objetivo común que persiguen las aproximaciones al diseño y desarrollo de sistemas distribuidos es el de facilitar la descripción de los mismos, refiriéndose a esta tarea como formalización de la arquitectura (o nivel de arquitectura) del sistema. El término *arquitectura* es ampliamente utilizado aun cuando no existe un consenso ni en su definición ni en su alcance. Esta falta de acuerdo no impidió que aparecieran numerosos lenguajes de descripción de arquitecturas, herramientas y entornos de trabajo asociados, modelos y patrones arquitecturales, técnicas de análisis y evaluación de arquitecturas, etc., y que, como consecuencia, se diversificara este ámbito en líneas de trabajo y metodologías superpuestas e incluso, a veces, contradictorias. Posteriormente desde varias organizaciones de estandarización surgieron recomendaciones, glosarios y guías de buenas prácticas para la descripción arquitectural que permitían establecer un marco de trabajo común y eliminar las divergencias y contradicciones entre propuestas [16-18].

A continuación se presentan los conceptos más relevantes descritos en estas recomendaciones y que serán utilizados en los apartados siguientes como marco terminológico de referencia. En la Figura 2.3 se muestran las relaciones entre estos conceptos.

- **Arquitecto** (*architect*): persona, equipo u organización responsable de la arquitectura del sistema.

- **Arquitectura** (*architecture o architectural model*): organización fundamental de un sistema representado por sus componentes (físicos o lógicos), la relación entre ellos y con el entorno, y los principios que guían su diseño y evolución.

- **Descripción arquitectural** (*architectural description*): colección de artefactos para documentar una arquitectura.

- **Diseño arquitectural** (*architecting*): actividades de definir, documentar, mantener, mejorar y certificar una adecuada implementación de una arquitectura. Con esta actividad se busca desarrollar conceptos del sistema de forma satisfactoria y eficiente, manteniendo su integridad a través del desarrollo así como de fases operacionales y evolutivas. Se busca además posibilitar la certificación de los sistemas construidos para su uso.

- **Participante, individuo o grupo interesado** (*stakeholder*): una persona, equipo u organización con intereses o implicación en un sistema.

- **Punto de vista** (*viewpoint*): especificación de las convenciones para construir y usar una vista. Un patrón o plantilla desde el cual desarrollar vistas individuales, estableciendo el propósito y audiencia para cada vista, así como las técnicas para su creación y análisis.

- **Sistema** (*system*): colección de componentes organizados para realizar una función o un conjunto de funciones.

- **Sistema legado** (*legacy system*): cualquier elemento (sistema de información, servicio, dispositivo…) que continúa en uso, generalmente porque todavía cubre ciertos requisitos de usuario, aun cuando su tecnología está obsoleta o existen métodos más eficientes para realizar la tarea que desempeña. Estos elementos heredados suponen un problema para la evolución e interoperatividad de los sistemas que los reutilizan puesto que deben ser adaptados a los nuevos protocolos o procesos de negocio.

- **Vista** (*view*): representación de un sistema completo desde la perspectiva de un conjunto relacionado de intereses o aspectos concretos.

Un *sistema* soporta necesidades en un entorno o *contexto* y existe para cumplir con una o más *misiones* en él. El sistema tiene uno o más *participantes* los cuales pueden desempeñar varios roles en la creación y uso de la descripción arquitectural como cliente, usuario, arquitecto, desarrollador, evaluador, etc. Cada participante tiene sus propios *intereses* relacionados con el desarrollo del sistema, su operación u otros. Una misión es un uso u operación para el cual uno o más participantes utilizan el sistema buscando alcanzar uno o más objetivos. Todo sistema tiene una *arquitectura* (sea conocida o no) que puede estar registrada por una *descripción arquitectural* (DA). Una DA está organizada en uno o más elementos constituyentes llamados *vistas*, cada una centrando uno o más intereses de los participantes del sistema. Un *punto de vista* establece las convenciones por las cuales una vista es creada, descrita y analizada; por tanto, una vista es conforme a un punto de vista, el cual determina el lenguaje para describir la vista, los métodos de modelado y las técnicas de análisis. Una DA selecciona uno o más puntos de vista a utilizar, generalmente basándose en los participantes y sus intereses. Una vista puede consistir en uno o más modelos.

Es necesario destacar que la actividad de diseño arquitectural es una fase más dentro del ciclo de vida del sistema y, dependiendo de la metodología elegida, tendremos una única fase de DA o iterativas descripciones (y sus correspondientes arquitecturas), evolucionando junto a las necesidades y participantes del sistema. Finalmente, la evaluación de una arquitectura no es más que la cuantificación del nivel de satisfacción de necesidades e intereses de los participantes para los cuales fue construida. Para obtener la conformidad con las mejores prácticas de descripción arquitectural, una DA debe constar de, al menos, los siguientes elementos:

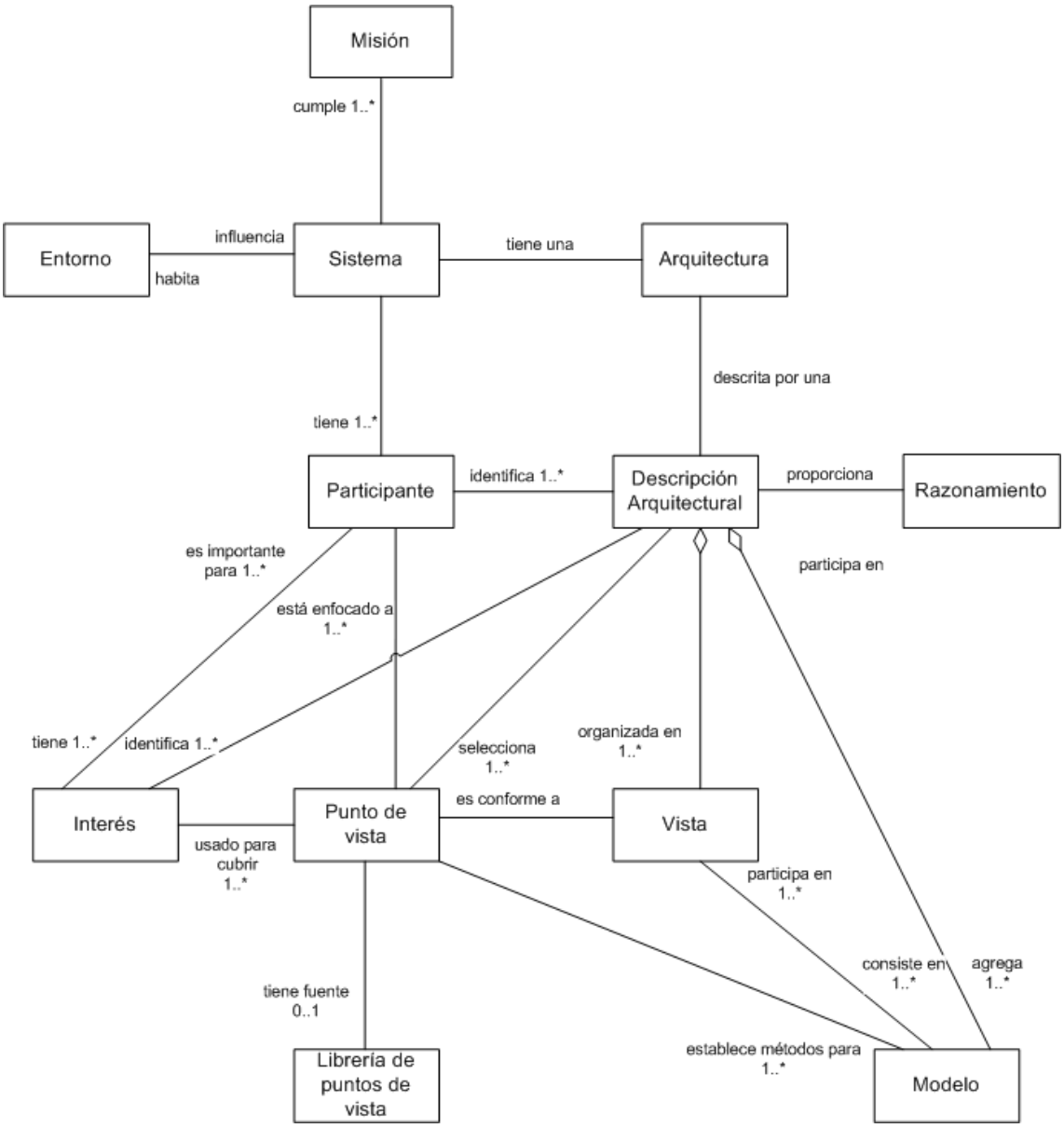

Figura 2.3. Marco conceptual de referencia de la norma ISO 42010 [17]

(a) Documentación arquitectural: fecha y estado, organización implicada en el desarrollo, historial de cambios, resumen, alcance, glosario y referencias.

(b) Identificación de participantes (usuarios, clientes, desarrolladores y mantenedores del sistema) e intereses (misión del sistema, adecuación con los intereses, factibilidad de construcción, riesgos, capacidad de evolucionar, ser desplegado y mantenido).

(c) Selección de puntos de vista arquitecturales

   a. Nombre, participantes objetivo, intereses a centrar, lenguaje, técnicas o métodos a emplear.

   b. Un razonamiento sobre la selección de cada punto de vista

(d) Vistas arquitecturales: identificador, representación del sistema construida con los lenguajes, métodos y técnicas del punto de vista correspondiente, e información de configuración.

(e) Consistencia entre vistas arquitecturales y registro de cualquier inconsistencia existente.

(f) Razonamiento sobre los conceptos arquitecturales utilizados.

Un fundamento clave compartido por la mayoría de recomendaciones es la separación de vistas en dos grupos según sean dependientes o independientes de tecnología. Las vistas que centran aspectos independientes de tecnología de una arquitectura pueden ser reutilizadas con distintas implementaciones tecnológicas sin más que desarrollar las vistas dependientes para la tecnología concreta. Esta descripción independiente de tecnologías permite enfrentar un importante obstáculo que sufren los sistemas de información en general, la portabilidad. En un campo en continua evolución como es el de las TICs constantemente surgen nuevas tecnologías que hacen obsoletas las anteriores. Por ello existe la necesidad de portar los procesos de negocio y aspectos de alto nivel de los sistemas a las nuevas tecnologías con facilidad. Contar con una DA independiente de tecnología permite reutilizar esfuerzos pasados y sólo centrarse en aquellos aspectos dependientes de la tecnología y que evolucionan con ésta.

Por otro lado, frente a una industria competitiva con un número masivo de desarrolladores de sistemas y dispositivos es crucial disponer de estándares que permitan la interoperatividad de los productos finales de manera que, con independencia de su origen, los distintos elementos que componen un sistema distribuido puedan colaborar entre sí. El diseño arquitectural de sistemas distribuidos permite especificar aspectos como los procesos de negocio que se deben llevar a cabo, los actores implicados, la información manejada, etc. Un desarrollador no tiene más que ceñirse a estas especificaciones para conseguir que su producto sea completamente interoperable con la arquitectura y el resto de elementos del sistema distribuido. Esta es una razón por la que estas iniciativas están siendo tan fuertemente apoyadas y utilizadas en la industria y la investigación.

Para obtener los beneficios presentados, el proceso de diseño arquitectural debe descansar en: una terminología común, una metodología de formalización/especificación del sistema y sus componentes, un razonamiento sobre cómo los diferentes elementos se ajustan y trabajan entre sí, y un conjunto de herramientas que soporten la metodología y faciliten la tarea del arquitecto. Existen diversos elementos (métodos, herramientas, etc.) dentro del ámbito de la formalización y especificación de arquitecturas de sistemas distribuidos. A lo largo de este capítulo y en lo que resta de trabajo se utilizarán las siguientes definiciones, las cuales se unen a las anteriormente descritas y concretan la terminología en cuanto a estos elementos.

- **Lenguaje de descripción** (*description language* o *modelling language):* conjunto de símbolos y de modos de disponerlos para modelar o describir (parte de) una entidad o sistema. Existen lenguajes

de descripción de propósito general como UML (*Unified Modeling Language*) [19], así como orientados a objetivos específicos como IDL (*Interface Description Language*) [20]. También el grado de abstracción de los lenguajes supone un aspecto diferenciador aunque estén centrados en el mismo ámbito de aplicación.

- **Marco arquitectural** (*architectural framework* o *conceptual framework for architectural description* o *frame of reference for architectural description*): estructura (o estructuras) conceptual que puede utilizarse para desarrollar un amplio rango de arquitecturas. Establece los términos y conceptos necesarios para llevar a cabo la descripción arquitectural de un sistema así como las relaciones entre conceptos y con términos que describen el mundo real. Un marco arquitectural es utilizado en el diseño arquitectural.

- **Metodología de desarrollo de arquitecturas** (*architecture development process*): secuencia formal de pasos con el propósito de desarrollar una arquitectura, cuyos componentes (físicos y lógicos) ofrezcan prestaciones que cubran las necesidades de los usuarios. Entre otras actividades se incluirán la especificación de los requisitos de los usuarios, el diseño de la arquitectura, la elección de tecnologías, etc.

- **Software de intermediación** (*middleware architecture*): capa con implementación tecnológica que proporciona un conjunto de servicios de apoyo al sistema relacionados con la gestión de datos comunes y procedimientos de relevancia capital para toda la organización. A través de sus servicios, el software de intermediación asegura la apertura e interoperatividad del sistema de información y una infraestructura operacional adecuada para la integración de sistemas legados y para el desarrollo de nuevos módulos. Tanto CORBA (*Common Object Request Broker Architecture*) [21] como las tecnologías de Servicios Web son consideradas software de intermediación.

En este apartado se realiza una descripción de los principales esfuerzos en el ámbito de la formalización de arquitecturas para sistemas distribuidos de propósito general. Además existen diversas iniciativas cuyo objetivo es la especificación de arquitecturas y servicios para el dominio sanitario. Los importantes esfuerzos que se están realizando en los últimos años por parte de organismos de estandarización para la especificación de estos servicios concretos (tanto de infraestructura como los relacionados con la práctica clínica) son el tema del último apartado del presente capítulo. Además se describirán los principales lenguajes de modelado sobre los que se apoyan los distintos estándares y métodos de formalización de arquitecturas. Debido a que los lenguajes de modelado son de propósito general, se indicarán las aportaciones y restricciones (si las hay) que cada estándar impone a los lenguajes.

3.2. ODP – Open Distributed Processing

El modelo de referencia ODP (RM-ODP) [5] es un esfuerzo de estandarización conjunto de ISO [4] e ITU [22] que persigue el desarrollo de estándares que permitan que los sistemas de procesado de

información distribuidos puedan ser explotados en entornos heterogéneos y bajo múltiples dominios administrativos. RM-ODP empieza a gestarse a finales de los años 80 ante la creciente necesidad de interconectar sistemas de procesamiento de información y desde entonces los requisitos de estos sistemas han ido haciéndose más numerosos y complejos y el modelo de referencia ha ido evolucionando y completándose paralelamente. En la actualidad RM-ODP consta de cuatro partes fundamentales, y una muestra de la relevancia de este modelo en la práctica científica e industrial actual es el hecho de que las Partes 2 y 3 fueron actualizadas en el 2009. Los cuatro estándares ISO en los que se basa el modelo son (también pueden encontrarse como recomendaciones ITU-T):

- ITU-T Rec. X.901 | ISO/IEC 10746-1 – Visión general (*Overview*): presenta los objetivos, motivación y aspectos fundamentales de ODP, así como una breve descripción de la arquitectura que propone el estándar.

- ITU-T Rec. X.902 | ISO/IEC 10746-2 – Fundamentos (*Foundations*): contiene la definición de los conceptos y el marco de trabajo necesarios para la descripción formal de sistemas de procesamiento distribuido. Introduce además los principios de conformidad con los estándares ODP y cómo deben ser aplicados.

- ITU-T Rec. X.903 | ISO/IEC 10746-3 – Arquitectura (*Architecture*): describe los lenguajes asociados a cada una de las vistas del modelo así como el conjunto de funciones de infraestructura que deben dar soporte al sistema de procesamiento distribuido y abierto a través de un conjunto de transparencias de distribución.

- ITU-T Rec. X.904 | ISO/IEC 10746-4 – Semántica arquitectural (*Architectural semantics*): contiene una formalización de los conceptos de modelado ODP mediante diferentes técnicas de descripción formal estandarizadas.

Además del marco de referencia que cubren estos cuatro estándares existe un conjunto de recomendaciones relacionadas con el procesamiento distribuido abierto que complementan el marco para facilitar la especificación, formalización y desarrollo de sistemas distribuidos basados en RM-ODP. Algunos de estos son el Lenguaje de empresa (*Reference Model: Enterprise language*) [23], la Función de intermediación (*Trading function*) [24], la Función de repositorio de tipos (*Type repository function*) [25], el Marco de nombrado (*Naming framework*) [26], y la norma UML4ODP - Uso de UML para las especificaciones de sistemas ODP (*Use of UML for ODP system specifications*) [27].

El alto valor de ODP es soportar el modelado, arquitectura y construcción de sistemas distribuidos y abiertos de forma transparente a la tecnología y desde varias perspectivas o vistas. La aplicación de éstas permite la separación de aspectos o intereses pertenecientes a varios participantes en el diseño y desarrollo del sistema. Por ejemplo, son muy diferentes las aproximaciones y el entendimiento que del mismo sistema tienen un director (que toma decisiones de negocio) y un técnico de redes (encargado

del hardware del sistema). Ambos pueden colaborar en el diseño y desarrollo del sistema desde sus dos puntos de vista diferentes de forma coherente y sin necesidad de adquirir conocimiento fuera de sus competencias puesto que el grado de entendimiento que cada uno tiene del sistema es suficiente para abordar aspectos concretos del mismo.

3.2.1. Fundamentos de RM-ODP

Los estándares desarrollados al amparo del modelo de referencia ODP (a partir de ahora, estándares ODP) promueven la especificación de sistemas que proporcionan portabilidad e interoperatividad de software, soportan la integración de sistemas con diferentes arquitecturas y recursos sin necesidad de soluciones ad-hoc costosas, acomodan la evolución de sistemas y cambios en tiempo de ejecución, federan a través de dominios separados técnica y administrativamente, incorporan calidad de servicio y conceptos de fallo, incluyen servicios de seguridad y ofrecen servicios de transparencia de distribución para la comunicación. Estos objetivos se logran mediante cuatro fundamentos estandarizados del modelo de referencia ODP.

1. Orientación a objetos.

El primer pilar de ODP es el paradigma de orientación a objetos, que permite la encapsulación y abstracción de comportamientos e información. Esta filosofía encaja con el concepto de servicio que apareció posteriormente como ha sido apuntado en la literatura [28], por ello los principios que guían el RM-ODP permiten acomodar estilos arquitecturales dispares como SOA o Event-OA. El comportamiento de un objeto es especificado como un conjunto de interfaces donde cada una puede representar los objetos como proveedores o consumidores de un servicio. Un objeto puede tener muchas interfaces y consecuentemente participar en la provisión o uso de muchos servicios. Una interfaz (a través de la cual dos o más objetos se comunican) no se especifica como una abstracción global e indivisible. En su lugar, tanto un cliente solicitando un servicio como un servidor proporcionándolo pueden tener perspectivas técnicas de la interfaz ligeramente diferentes. Los conceptos del modelo de comunicación ODP declaran cómo los detalles técnicos de dichas perspectivas pueden ser mapeados entre sí en el proceso de vinculación (*binding*) y crear así un canal de comunicación entre las interfaces del cliente y el servidor. La separación entre interfaces facilita la distribución, adaptabilidad y vinculación a través de fronteras organizacionales.

2. División de la especificación del sistema ODP en cinco puntos de vista independientes pero relacionados para simplificar la descripción de los sistemas.

El segundo aspecto clave del RM-ODP es la definición de cinco puntos de vista los cuales permiten especificar vistas como proyecciones separadas de la descripción completa del sistema. Cada especificación de una vista es completa y consistente en sí misma pero sólo considera aquellos aspectos del sistema válidos para el correspondiente punto de vista. Los cinco puntos de vista son:

- Punto de vista de la empresa (*enterprise viewpoint*): identifica actividades y responsabilidades del sistema, siendo una actividad un intercambio (o secuencia de intercambios) de información. Se identifica a su vez el sistema, su entorno y la comunicación necesaria entre ambos. Este punto de vista se corresponde con el modelo arquitectural de negocio aunque sin ninguna conexión a servicios elementales de la arquitectura o aspectos tecnológicos necesarios para soportarlo.

- Punto de vista de la información (*information viewpoint*): identifica entidades lógicas de información, sus contenidos, repositorios y objetos responsables del flujo de información en los sistemas. Se centra en la semántica de la información y no en cómo se almacena (en qué soporte o formato).

- Punto de vista computacional (*computational viewpoint*): captura el comportamiento del sistema (esto es, la abstracción de cómo se hacen las cosas). Presenta el sistema como una composición de objetos lógicos para cada uno de los cuales se definen sus interfaces. Si la interfaz implica operaciones se describen a su vez los parámetros lógicos. Si la interfaz implica flujos, cada componente del flujo de datos se asocia con descripciones de protocolos lógicos. Este punto de vista muestra explícitamente el potencial de distribución.

- Punto de vista de ingeniería (*engineering viewpoint*): identifica los servicios de infraestructura necesarios para el funcionamiento del sistema. El modelo de infraestructura del RM-ODP identifica un conjunto de servicios básicos, distribuidos y globales que deberían estar disponibles para cada nodo en el sistema global. Estos servicios básicos facilitan la transparencia de comunicación e incluyen la invocación de operaciones, transferencia de datos continuos como flujos (*streams*), función de *trading* (proporciona un repositorio de oferta de servicios), repositorios de tipos (provee de reglas de estructuración restricciones para metainformación y elementos de servicio gestionados por la plataforma), etc.

- Punto de vista de tecnología (*technology viewpoint*): muestra cómo se implementan los servicios del sistema y otros componentes requeridos a través de configuración hardware y software.

Al margen de la especificación de cada vista, se debe establecer cómo las especificaciones se mapean entre sí, debido a que cualquier entidad del mundo real representada por un término en una vista debe satisfacer también los requisitos impuestos por la ocurrencia del término correspondiente en otras vistas. Este hecho remarca una vez más que, aunque las vistas son especificaciones completas, todas se refieren al mismo sistema, por eso de la necesidad de establecer correspondencias entre ellas. Las correspondencias más relevantes son aquellas que se dan entre vistas consecutivas (empresa e información o ingeniería y tecnología). En la Figura 2.5 se indican las correspondencias de interés entre vistas. Cambios en la especificación de una vista durante el diseño puede llevar a inconsistencias con otras vistas, y las correspondencias declaradas pueden ser utilizadas para propagar y gestionar estos cambios.

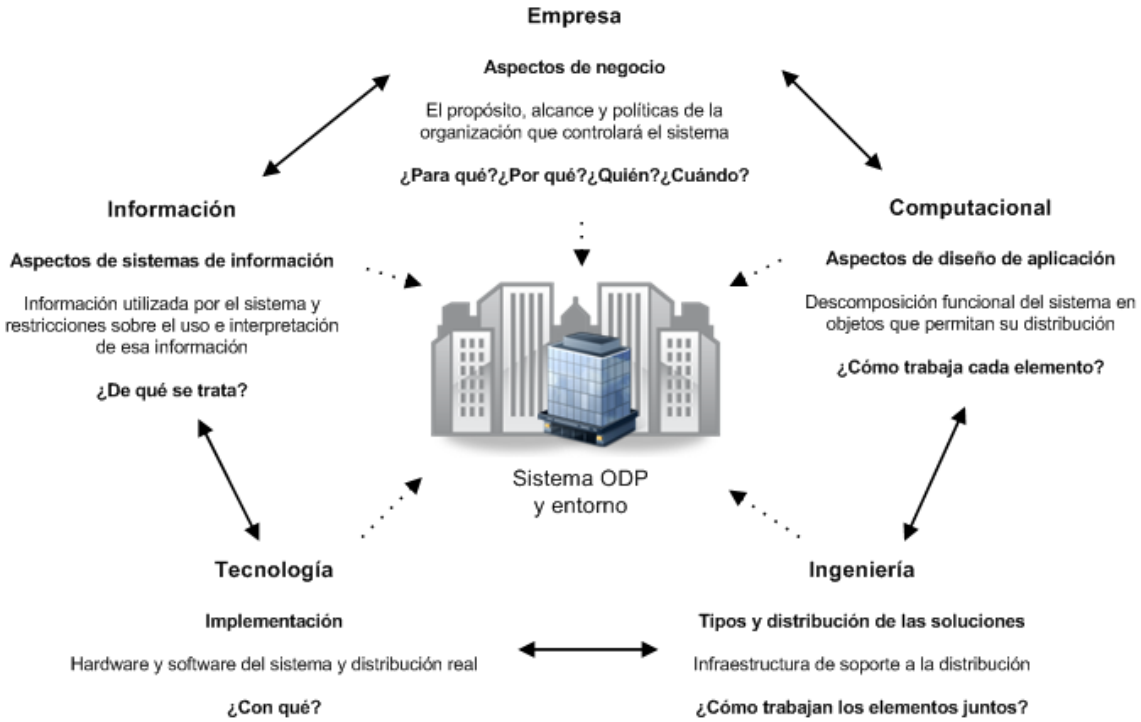

Figura 2.4. Las vistas del modelo de referencia ODP

3. Definición de un conjunto de conceptos generales para expresar la especificación de las vistas.

El tercer aspecto fundamental del RM-ODP consiste en los conceptos que permiten la especificación del sistema en vistas. Por ejemplo, se pueden encontrar los conceptos de comunidad (*community*), dominio (*domain*) y federación (*federation*). Pueden utilizarse para estructurar objetos y pueden ser considerados conceptos estáticos (en tiempo de diseño) o dinámicos (en tiempo de operación). Una comunidad es una configuración de objetos con un objetivo común. Un dominio <X> se define como un conjunto de objetos, cada uno de los cuales está relacionado con un objeto control a través de una relación <X>. Por ejemplo, un dominio de intermediación (trading) está formado por todos los objetos conocidos por el objeto intermediario (*trader*). Por último una federación <X> es una comunidad de dominios <X>.

4. Modelo para la infraestructura de soporte a través de la provisión de transparencias de distribución.

El último aspecto clave del RM-ODP son las transparencias de distribución que permiten ocultar las complejidades asociadas con la distribución del sistema a las aplicaciones no directamente implicadas en estas tareas. De esta forma las aplicaciones no tienen que contemplar esas capacidades en su diseño puesto que van a ser provistas por la arquitectura ODP. Entre las transparencias de distribución están la de acceso, de fallos, de localización, de migración, de replicación, etc. Para conseguir la completa transparencia de distribución han sido identificadas funciones y estructuras en los estándares ODP de las cuales sólo las siguientes han sido completamente definidas a través de estándares: la función de intermediación [24] que media entre la publicación y el descubrimiento de interfaces; la función de repositorio de tipos [25] que gestiona un repositorio de especificaciones de tipos y relaciones entre

ellos, y posee una interfaz para cada especificación de tipo que almacena; y el marco de nombrado [26], el cual define un marco de trabajo general para el nombrado relativo a contexto y funciones asociadas.

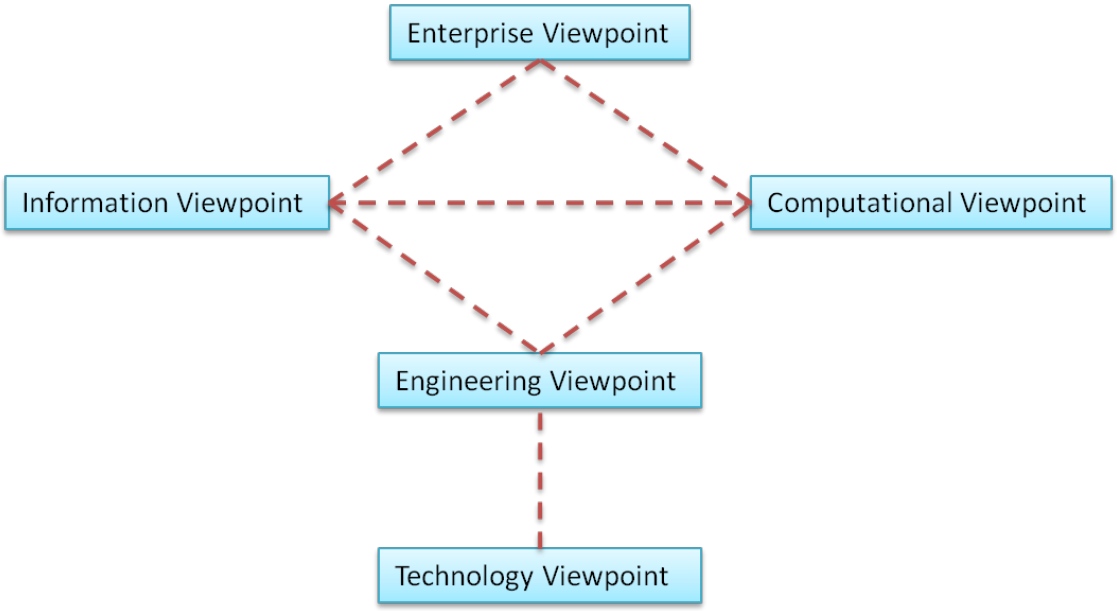

Figura 2.5. Correspondencias entre vistas del modelo de referencia ODP

Finalmente es necesario destacar que el RM-ODP plantea mecanismos para testear la conformidad de las implementaciones desarrolladas, esto es, comprobar la relación que existe entre la especificación y la implementación real del sistema y si esta última satisface los requisitos concretos expresados en la especificación y es coherente con las vistas ODP del sistema. La evaluación de la conformidad proporciona una medida de la calidad, principalmente en la fase en que se realiza la implementación. Cuando la conformidad de una realización de una especificación ODP se evalúa mediante pruebas de conformidad, su comportamiento se estudia (aplicándole estímulos y supervisando todo evento resultante) en puntos de interacción específicos. Estos puntos se denominan “de conformidad” y RM-ODP define cuatro tipos: programático (en el que puede establecerse una interfaz vinculada a un lenguaje de programación que permite el acceso a una función), perceptual (punto en el que hay alguna interacción entre el sistema y el mundo físico), de interfuncionamiento (donde puede establecerse una interfaz para permitir la comunicación entre dos o más sistemas) y de intercambio (en el que puede introducirse un medio físico externo de almacenamiento).

3.2.2. Otras consideraciones

Habiendo descrito los fundamentos básicos de este marco de referencia cabe destacar un par más de consideraciones. Por un lado, para la formalización de las arquitecturas según el RM-ODP pueden usarse las herramientas de modelado UML. En concreto existe un perfil de UML para ODP que está estandarizado en la norma ITU-T Rec. X.906 | ISO 19793 [27]. Los principios generales de esta norma

serán especificados en un apartado posterior donde a su vez se indicarán los mecanismos utilizados para establecer la consistencia entre vistas y la conformidad de éstas con el sistema real.

Finalmente es necesario mencionar que, si bien los conceptos y definiciones del modelo de referencia ODP han influenciado notablemente el campo de la formalización de arquitecturas de sistemas distribuidos abiertos, algunos de estos conceptos o sus definiciones han sido objeto de críticas, discusión y debate en la literatura [29-31]. Algunos de estos puntos de discusión han sido:

- RM-ODP presenta un marco arquitectural muy completo en cuanto a la especificación de sus vistas y las correspondencias entre ellas pero aboga por flexibilizar el uso de metodologías y procesos de desarrollo. Como se verá más adelante RM-ODP puede ser combinado con procesos de desarrollo de sistemas aunque, como ha sido sugerido en la literatura, ninguna metodología se ajustará a los fundamentos de RM-ODP tan adecuadamente como lo podría hacer una dedicada a este modelo de referencia.

- A la luz de las propuestas del OMG (*Object Management Group*) [32] para la transformación entre modelos, algunos autores critican la falta de un metamodelo de conceptos de ODP de manera que se puedan aplicar tales técnicas y desarrollar mecanismos de transformación entre vistas. En este sentido el perfil de UML para modelado con ODP ha venido a posibilitar la aplicación de estas técnicas.

3.3. MDA - Model-Driven Architecture

La arquitectura conducida por modelos (o por sus acrónimos en inglés, MDA) [33] es una propuesta basada en estándares para el desarrollo de sistemas software definida por el OMG y publicada en el año 2003. Esta iniciativa se basa en dos claves principales:

1. La separación de la funcionalidad de negocio del sistema de los detalles de cómo éste utiliza las capacidades de la plataforma tecnológica subyacente para alcanzar dicha funcionalidad. De este punto derivan los principios de portabilidad, interoperatividad y reusabilidad que obtienen los sistemas desarrollados a través de esta herramienta.

2. El incremento en la importancia de los modelos y la posibilidad de establecer transformaciones entre ellos. Así los modelos están presentes en todo el ciclo de desarrollo del sistema y permiten dirigir las fases de entender el sistema, diseñarlo, construirlo y desplegarlo, así como las tareas de operación, mantenimiento y modificación.

MDA también hereda el concepto de punto de vista del estándar IEEE 1471 y lo modifica para definir un punto de vista MDA como una técnica de abstracción de sistemas que utiliza un conjunto de conceptos arquitecturales y reglas de estructuración con el objeto de centrarse en aspectos particulares del

sistema. Siguiendo este concepto, MDA especifica tres puntos de vista: independiente de computación, independiente de plataforma y dependiente de plataforma.

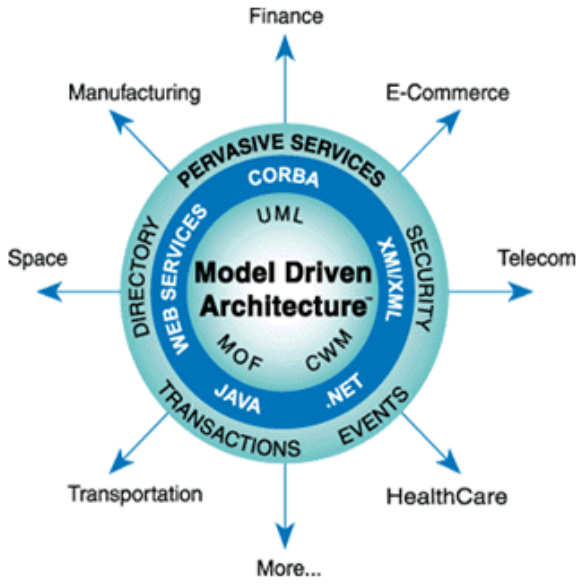

Figura 2.6. Herramientas, tecnologías y dominios relacionados con MDA

En primer lugar tenemos el punto de vista independiente de computación (*computation independent viewpoint*) el cual se centra en el entorno y los requisitos a los que está sujeto el sistema. La representación del sistema desde este punto de vista se denomina modelo independiente de computación (*computation independent model* o CIM) y se compone de dos o más modelos UML. A continuación se introduce el punto de vista independiente de plataforma y su correspondiente modelo (*platform independent model* o PIM). Este punto de vista busca mostrar los aspectos de la especificación que no cambian de una plataforma a otra, es decir, modelar el sistema desde un alto nivel de abstracción independiente de cualquier implementación tecnológica. De esta forma se centra principalmente en la operación del sistema (procesos de negocio) pero sin especificar cómo se lleva a cabo. En este punto de vista se puede utilizar un lenguaje de modelado de propósito general como UML o específico del área en el que el sistema será utilizado. Por último tenemos el punto de vista específico de plataforma el cual combina el anterior con detalles del uso de una plataforma específica por parte del sistema. El modelo específico de plataforma (*platform specific model* o PSM) es una representación de este punto de vista, es preciso y completo, conforme a las restricciones de una plataforma concreta y está orientado a expertos en tecnología. Un PSM permite especificar el sistema en términos de los constructores de implementación disponibles en una tecnología de implementación específica. Para ello hace uso del modelo de plataforma (*platform model* o PM) que proporciona un conjunto de conceptos técnicos representando las diferentes partes que forman una plataforma y los servicios que proporciona (así por ejemplo el PM de CORBA es el *CORBA Component Model* o CCM). Un PM también especifica los requisitos en la conexión y uso de las partes de la plataforma y las conexiones necesarias entre una aplicación y la plataforma.

El definir la propuesta MDA como "basada en estándares" tiene como significado que está asociada tanto a estándares desarrollados por el OMG como a un conjunto publicado por otras organizaciones (como el IEEE 1471). Estos estándares comprenden principalmente los relacionados con lenguajes de modelado donde el más reconocido desarrollado por el OMG es el UML incluyendo el lenguaje de restricciones OCL (*Object Constraint Language*) [34]. A través de los lenguajes de modelado estandarizados los modelos en MDA son especificados formalmente lo que permite extraer todo el potencial de transformaciones entre ellos. Por eso, los lenguajes utilizados necesitan tener definiciones formales para que las herramientas puedan realizar las transformaciones de forma automática. El OMG desarrolló como meta-lenguaje el MOF (*Meta Object Facility*) [35] utilizado para describir otros lenguajes.

Una herramienta de modelado para MDA asiste al proceso de transformación del CIM (expresado en lenguaje UML) capturando los procesos de negocio del sistema. A este modelo se le añadirá información para obtener el PIM y posteriormente, tomando un conjunto de decisiones de diseño y usando perfiles estándar del OMG, se pasará del PIM a uno o varios PSM. Tanto este paso como el anterior no tienen por qué estar constituidos por una única transformación; de hecho lo normal es que se tengan que realizar múltiples transformaciones intermedias entre diferentes niveles de abstracción. Finalmente la herramienta genera el código para la plataforma específica seleccionada, paso que suele ser sencillo debido a que PSM ya se ajusta muy bien a la tecnología concreta. En este proceso no sólo los modelos MDA deben estar especificados formalmente sino también las transformaciones entre ellos, de manera que las herramientas puedan realizarlas de forma automática. Las definiciones de transformaciones se formalizan a través del lenguaje QVT (*Query, Views and Transformations*) [36].

Por último, cabe destacar que MDA no establece ningún tipo de metodología y menos aún obliga al modelado en cascada (como se podría intuir del razonamiento anterior). La propuesta del OMG establece claramente que lo ideal del proceso es que exista realimentación entre las diferentes etapas. Aunque no especifica cuál sería el método más apropiado sí aconseja que sea cíclico en algún sentido.

3.4. TOGAF – The Open Group Architecture Framework

TOGAF [37] es un marco de trabajo arquitectural que proporciona métodos y herramientas para asistir y facilitar la aceptación, producción, uso y mantenimiento de una arquitectura de empresa. Está basado en un modelo de proceso iterativo soportado por las mejores prácticas en este campo y un conjunto reutilizable de activos arquitecturales. La primera versión de TOGAF surgió en 1995 de la mano de *The Open Group* [38] como continuación de iniciativas anteriores y la versión actual (número 9) fue publicada en el año 2009. TOGAF consta de tres pilares fundamentales: el Método de desarrollo de arquitecturas (ADM), el Espectro continuo de empresa y el Repositorio de arquitecturas.

3.4.1. Método de desarrollo de arquitecturas (*ADM – Architecture Development Method*)

Esta metodología es el núcleo de TOGAF y consiste en una propuesta paso a paso que guía el proceso de de desarrollo y uso de una arquitectura de empresa. Es un proceso iterativo que tiene en cuenta los requisitos de negocio en todos los pasos de la metodología. Permite el diseño y desarrollo de arquitecturas pero también su prueba y realización. Para ello a lo largo del proceso se establecen entregables, artefactos y bloques constructivos que van permitiendo el análisis y la evaluación de la arquitectura de forma continua. El ADM define una secuencia de diez fases para, a partir de los requisitos de negocio, establecer el marco de trabajo, desarrollar el contenido de la arquitectura y llevar a cabo la realización de la misma. Este proceso incluso tiene en cuenta los posibles planes de migración necesarios para sustituir los sistemas actuales de la empresa a la arquitectura obtenida con TOGAF. Cada fase define unos objetivos a realizar, así como sus entradas y salidas, y el desarrollo de cada fase se lleva a cabo mediante un conjunto de pasos o actividades específicos. El proceso del ADM (Figura 2.7) consta de las siguientes fases:

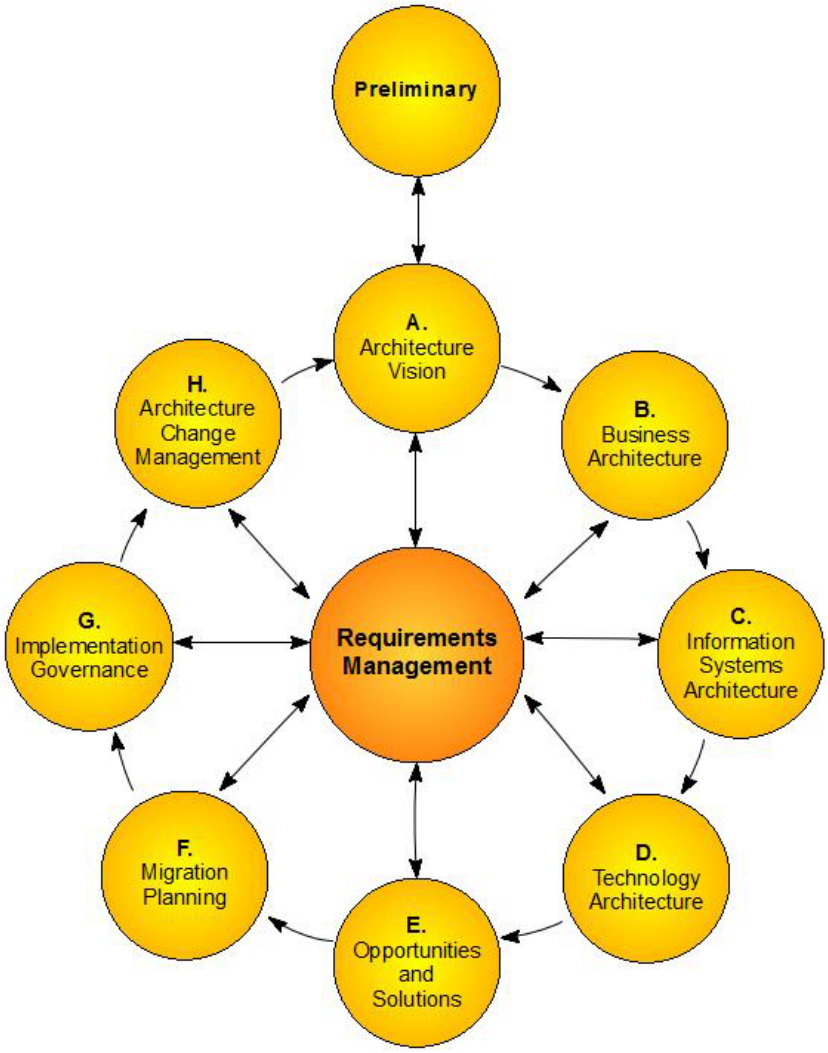

Figura 2.7. Fases del proceso TOGAF ADM [37]

(a) **Fase preliminar** (*Preliminary Phase*): describe las actividades necesarias para preparar las directivas de negocio para la nueva arquitectura, incluyendo la definición de un marco de trabajo de arquitectura específico de la organización y los principios establecidos por la misma.

(b) **Fase A. Visión de arquitectura** (*Architecture Vision*): describe y organiza el ciclo de desarrollo de la arquitectura e incluye información sobre su alcance, los participantes implicados, objetivos de negocio, etc. En esta fase se crea la visión de arquitectura que es una descripción a alto nivel de la arquitectura objetivo a la que se aspira. Permite marcar una serie de objetivos a cumplir en el resto de fases para alcanzar esa visión.

(c) **Fase B. Arquitectura de negocio** (*Business Architecture*): identifica los requisitos clave de negocio y describe el desarrollo de la arquitectura de negocio base en consonancia con la visión de arquitectura de la fase anterior. La arquitectura resultante de esta fase incluye la estrategia de negocio, directivas, organización e información de los procesos de negocio claves, además de las interacciones entre estos conceptos.

(d) **Fase C. Arquitecturas de sistemas de información** (*Information Systems Architectures*): esta fase describe el desarrollo de las arquitecturas de sistemas de información, incluyendo el desarrollo de las arquitecturas de datos y aplicación (*Data and Application Architectures*). Identifica y define las consideraciones en el dominio de los sistemas de datos y aplicaciones. La arquitectura de datos define los tipos y fuentes de datos necesarios mientras que la arquitectura de aplicación especifica los tipos de sistemas de aplicación (entendido el término aplicación como un grupo de capacidades lógicas) necesarios para procesar los datos y soportar los procesos de negocio.

(e) **Fase D. Arquitectura tecnológica** (*Technology Architecture*): centrada en mapear la arquitectura de aplicación a componentes.

(f) **Fase E. Oportunidades y soluciones** (*Opportunities & Solutions*): revisa los entregables obtenidos en las fases anteriores y organiza la implementación inicial de la arquitectura.

(g) **Fase F. Plan de migración** (*Migration Planning*): especifica un conjunto secuencial y detallado de arquitecturas de transición para la migración desde la arquitectura implantada actualmente en la empresa a la nueva propuesta desarrollada.

(h) **Fase G. Directivas de implementación** (*Implementation Governance*): proporciona las directivas de implementación.

(i) **Fase H. Gestión de cambios de la arquitectura** (*Architecture Change Management*): establece los procedimientos para gestionar cambios en la nueva arquitectura.

(j) **Gestión de requisitos** (*Requirements Management)*: las actividades relacionadas con la gestión de requisitos de arquitectura son transversales a todo el ADM.

3.4.2. El Espectro continuo de empresa (*Enterprise Continuum*) y el Repositorio de arquitectura (*Architecture Repository*)

TOGAF permite el desarrollo de arquitecturas desde cero a través de su ciclo ADM pero esto sólo es una parte del potencial de este marco de trabajo. Para facilitar el desarrollo de arquitecturas TOGAF permite la reutilización de componentes o activos de arquitecturas ya desarrolladas (descripciones, modelos, patrones de diversas fuentes, etc.) y su utilización dentro del ciclo ADM para la construcción de una nueva arquitectura. El Repositorio de arquitectura TOGAF almacena estos activos pero es el Espectro continuo de empresa el que proporciona el marco de trabajo y contexto para soportar la herencia de dichos activos. Este mecanismo permite describir cómo soluciones genéricas (los activos) pueden ser heredadas y especializadas para satisfacer los requisitos de una organización particular. El Espectro es una herramienta para categorizar material arquitectural y modelos de referencia disponibles en la industria. Estos elementos evolucionarán desde lo que se denominan arquitecturas fundamentales y genéricas a las arquitecturas específicas de organización que son las que se pretenden desarrollar con el ciclo ADM. La implementación práctica del Espectro tomará la forma de un Repositorio de arquitectura e incluirá arquitecturas de referencia, modelos y patrones.

El Repositorio de arquitectura puede ser utilizado para almacenar diferentes tipos de soluciones arquitecturales en diferentes niveles de abstracción y creadas usando el ciclo ADM. De esta forma, TOGAF facilita la cooperación entre participantes y desarrolladores a diferentes niveles. A través de este mecanismo y el Espectro continuo, se anima a la herencia de recursos y activos arquitecturales existentes en el desarrollo de la arquitectura específica para una organización. El ADM describe un proceso que opera a múltiples niveles de la organización y que produce salidas que se almacenan en el Repositorio de arquitectura. El Espectro es el encargado de darle una coherencia a los modelos y activos arquitecturales, especificando los bloques constructivos, las relaciones entre ellos, así como las restricciones o requisitos que se plantean en el ciclo de desarrollo de la arquitectura.

3.5. Otros marcos de trabajo

Existen numerosas metodologías y marcos de trabajo para la formalización de arquitecturas al margen de RM-ODP, MDA y TOGAF. A continuación hacemos un resumen de las más relevantes si bien se podrá comprobar que o están limitadas a entornos muy específicos de trabajo o no son tan completas ni maduras como las mencionadas previamente.

3.5.1. Zachman Framework

Este marco de trabajo surge en el año 1987 y puede considerarse el precursor del resto de metodologías [16]. Aparece en un momento tecnológico en el que los sistemas de información distribuidos masivos

son una mera especulación y están lejos de ser desplegados en la realidad. Pese a ello se considera que una futura descentralización de los sistemas sin una estructura formal conducirá al malfuncionamiento de los mismos y J.A. Zachman entiende que la distribución requiere una estructura (utiliza ya el término arquitectura) que mantenga la cohesión de los componentes distribuidos.

Esta primera iniciativa es un marco de trabajo y no una metodología de planificación estratégica que establezca los pasos a seguir para el desarrollo de sistemas distribuidos. Partiendo de una analogía con los productos que se obtienen en el desarrollo de un proyecto arquitectural real, es decir, un edificio, enumera los productos arquitecturales de los sistemas de información y llega a una clasificación de conceptos y especificaciones. Esta propuesta sigue la filosofía de descomposición de la arquitectura completa en perspectivas. En concreto posee cinco perspectivas: una de descripción a alto nivel (alcance, propósito, etc.), tres más orientadas a cada uno de los actores implicados (propietario, diseñador y constructor) y una más con detalles técnicos cercanos al producto final. Estas representaciones no van en cascada ni siguen un orden concreto de realización puesto que no están configuradas con diferentes niveles de detalle sino que centran diferentes perspectivas de la misma arquitectura, es decir, tienen diferente naturaleza.

Paralelamente a la descripción de la arquitectura desde las perspectivas de los diferentes participantes, el marco de trabajo Zachman considera diferentes tipos de descripciones para el mismo producto independientemente del actor implicado. Adelantándose al concepto de vista, se indicó que estas descripciones son aisladas aunque no dejan de estar relacionadas puesto que se refieren al mismo producto. Se definieron tres descripciones principales:

- WHAT (modelo de datos): centrado en la información intercambiada entre componentes de la arquitectura.

- HOW (modelo de proceso): funciones de negocio y actividades que realizan los componentes de la arquitectura.

- WHERE (modelo de red): referido a aspectos y decisiones tecnológicas concretas.

En resumen el marco de trabajo Zachman se basa en dos premisas:

1. Existe un conjunto de representaciones arquitecturales producidas en el proceso de construir un sistema complejo representando las diferentes perspectivas de los participantes.

2. El mismo producto puede estar descrito para propósitos separados en formas diferentes y resultando en tipos de descripciones distintos.

Como resultado, cada descripción (2) tiene un punto de vista para cada uno de los participantes implicados (1), lo que nos lleva a la tabla del marco de trabajo de la Figura 2.8 donde las columnas son las descripciones y las filas las aproximaciones de cada actor implicado. La iniciativa Zachman no impone

herramientas ni lenguajes de modelado para la formalización de la arquitectura, sólo el marco de trabajo arquitectural.

	DATA *what*	FUNCTION *How*	NETWORK *Where*
Objective Scope *Contextual* *Role: Planner*	List of Things Important in the Business	List of core Business Process	List of Business Locations
Enterprise Model *Conceptual* *Role: Owner*	Conceptual Data/ Object Model	Business Process Model	Business Logistics System
System Model *Logica* *Role: Designer*	Logical Data Model	System Architecture Model	Distributed Systems Architecture
Technology Model *Physical* *Role: Builder*	Physical Data/ Class Model	Technology Design Model	Technology Architecture
Detailed Representations *out of Context* *Role: Programmer*	Data Definations	Program	Network Architecture
Functioning Enterprise *Role: User*	Usable Data	Working Function	Usable Network

Figura 2.8. Cuadro resumen del marco de trabajo de Zachman

3.5.2. SOMF - Service Oriented Modeling Framework

SOMF [39] es un marco de trabajo para el modelado integral, independiente de tecnología de sistemas orientados a servicios. Cubre el diseño y la arquitectura de negocio para pequeños y grandes proyectos pudiendo particularizarse los resultados para tecnologías específicas. Su orientación a servicio se materializa en la representación de todos los activos software de la organización como servicios, los cuales clasifica en: servicio atómico (entidad de software indivisible), servicio compuesto (aplicación que aglutina o utiliza dos o más componentes software) y clúster de servicios (grupo de servicios que proporcionan una solución a un problema organizacional específico). Los servicios son descritos a través de modelos y SOMF se basa en la transformación entre éstos para particularizarlos para tecnologías concretas. Posee su propio lenguaje de modelado y describe un conjunto de diez modelos diferentes. SOMF está especialmente centrado en la implementación de los paradigmas de diseño SOA y Cloud computing y para el desarrollo software ofrece seis disciplinas de trabajo:

1. Conceptualización orientada a servicios: descripción de requisitos

2. Arquitectura conceptual

3. Análisis y descubrimiento de servicios

4. Integración de negocio orientada a servicios

5. Diseño orientado a servicios, centrado en la relación entre servicios, el diseño de la composición de servicios y las transacciones entre ellos

6. Arquitectura lógica

3.5.3. ARIS - Architecture of Integrated Information Systems

ARIS [40] ofrece métodos para analizar procesos de negocio y conformar una vista integrada del diseño, gestión, flujo de trabajo y procesamiento de aplicaciones. Aporta un marco de trabajo metodológico y una herramienta de modelado de procesos de negocio denominada *ARIS Tool Set* así como un lenguaje de modelado propio, el *Event-driven Process Chains* (EPC). Esta iniciativa está orientada a procesos lo que se traduce en que modela las arquitecturas para describir procesos de negocio. La Figura 2.9 muestra el modelo de vistas de ARIS, el cual se separa en cuatro grandes bloques: organización, datos, control y procesos. Para cada uno de ellos se definen tres niveles: conceptual, de procesamiento de datos o funcional y el nivel de implementación.

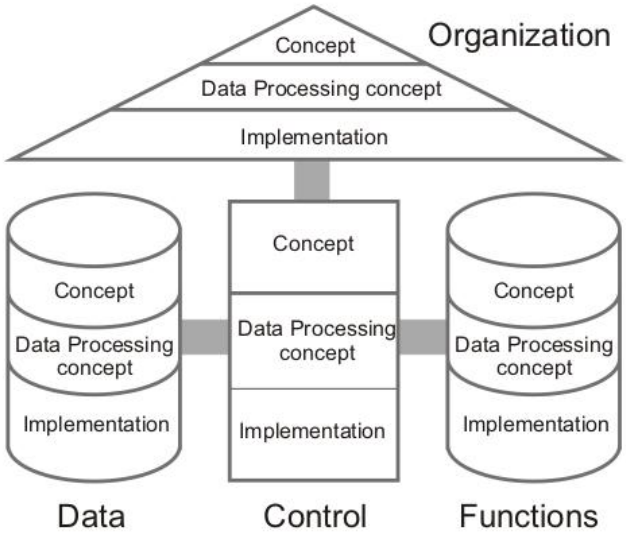

Figura 2.9. Modelo de vistas del marco de trabajo ARIS

3.5.4. Praxeme

Definido en el año 2004 (y con la última actualización publicada en el año 2010), Praxeme [41] se define como un método abierto que busca apoyarse en estándares y mejores prácticas para construir un marco de trabajo de formalización de arquitecturas. Define una estructura básica sobre la que integrar las

distintas contribuciones externas como los estándares UML, MDA, BPM (*Business Process Management*), etc. Su finalidad última es establecer una referencia metodológica común para toda la industria a través del uso extensivo de técnicas de modelado orientado a objetos, el reconocimiento de los beneficios del modelado semántico, la aceptación de la complejidad de los sistemas, el conocimiento de la dimensión económica y su influencia en el diseño de sistemas, y el reconocimiento de la importancia de la dimensión humana y de comunicación implicada en los sistemas.

Praxeme se basa en un conjunto de aspectos que centran el sistema y se materializan en diferentes puntos de vista. Esta iniciativa tiene muy en cuenta las relaciones entre aspectos las cuales permiten derivar y transformar elementos entre aspectos y mantener la consistencia entre ellos y con el sistema completo. Los aspectos que define Praxeme son:

- aspecto semántico: busca definir formalmente el conocimiento básico y nuclear del sistema.
- aspecto pragmático: se centra en los roles, procesos, casos de uso, etc.
- aspecto geográfico: restricciones y decisiones sobre la localización de actividades.
- aspecto lógico: decisiones estructurales independientes de tecnología.
- aspecto técnico: decisiones estructurales dependientes de tecnología.
- aspecto físico: define las relaciones entre los elementos hardware y software.

Utiliza como lenguaje de modelado el estándar UML y recomienda complementarlo con otros como OCL. Paralelamente, esta iniciativa está alineada con MDA puesto que sus herramientas y técnicas pueden ser adaptadas para la transformación entre modelos de un aspecto a otro.

3.5.5. FEAF - Federal Enterprise Architecture Framework

Esta iniciativa surgió en 1999 por parte de los organismos oficiales de Estados Unidos como un mecanismo para establecer un marco de trabajo muy orientado a la realidad de los estados federados y las agencias que operan en ellos [42]. Comprende un conjunto de modelos de referencia:

1. Modelo de referencia de ejecución (*Performance reference model*, PRM): centrado en los procesos de negocio a nivel estratégico establece un lenguaje común para el análisis de los mismos. Su objetivo último es facilitar la toma de decisiones de negocio.
2. Modelo de referencia de negocio (*Business reference model*, BRM): vista funcional de las líneas de negocio incluyendo operaciones y servicios que ofrecen las agencias a los ciudadanos.

3. Modelo de referencia de componentes de servicio (*Service component reference model*, SRM): marco de clasificación de servicios en función de los procesos de negocio que soportan y sus objetivos.

4. Modelo de referencia técnico (*Technical reference model*, TRM): categoriza estándares y tecnologías para soportar y permitir la disponibilidad de componentes de servicio y capacidades.

5. Modelo de referencia de datos (*Data reference model*, DRM): marco de trabajo basado en estándares que permite la reutilización e intercambio de información a través de descripciones estándar y capacidades de descubrimiento.

3.5.6. DoDAF - Department of Defense Architecture Framework

Otra iniciativa perteneciente a los Estados Unidos (en concreto al Departamento de Defensa) es DoDAF [15] la cual proporciona guías y reglas para desarrollar, representar y entender las arquitecturas de este departamento. DoDAF establece un proceso de desarrollo de arquitecturas estructurado en seis pasos a alto nivel que simplifica la tarea del arquitecto:

1. Determinar el uso intencionado de la arquitectura: incluyendo requisitos de los participantes, puntos críticos, propósito, puntos de decisión, etc.

2. Determinar el alcance de la arquitectura: definiendo los límites operacionales, geográficos, funcionales, temporales y tecnológicos.

3. Determinar los datos requeridos para el desarrollo de la arquitectura: niveles de detalle, metadatos asociados, etc.

4. Recoger, organizar, relacionar y almacenar los datos de la arquitectura: modelos de actividad, de datos, dinámicos, etc.

5. Realizar análisis sobre el cumplimiento de los objetivos de la arquitectura: se analiza la capacidad de la arquitectura, los fallos, la capacidad de interoperación, etc. Este análisis puede conducir a realizar cambios en el paso 3.

6. Documentar resultados: en forma de vistas y productos de la arquitectura.

Los productos resultados del desarrollo de una arquitectura en DoDAF se agrupan en: vista operacional (*Operational View*, centra tareas y actividades de nodos, así como la información intercambiada), vista de servicios y sistemas (*Systems and Services View*, sistemas, servicios y funcionalidad de interconexión), vista de estándares técnicos (*Technical Standards View*, estándares y reglas de implementación) y vista completa (*All-View*, que reúne aspectos incluidos en las tres vistas anteriores como contexto, alcance, etc.).

3.5.7. MoDAF - Ministry of Defense Architectural Framework

En el año 2005, el Ministerio de Defensa del Reino Unido publicó su propio marco de trabajo arquitectural con el fin de asistir a la toma de decisiones [43]. MoDAF abstrae la información esencial de la complejidad subyacente y la presenta de manera coherente. Define la forma en que se debe representar una arquitectura empresarial y permite que los participantes implicados se centren en áreas de interés específicas sin perder de vista el conjunto global. Para ello divide el problema en piezas categorizadas bajo diferentes puntos de vista en función de la perspectiva de que se trate. MoDAF hereda y mantiene la compatibilidad con DoDAF aunque añade dos puntos de vista a los especificados por la iniciativa americana:

- Punto de vista estratégico (*Strategic viewpoint*, StV): analiza y determina cómo evolucionan las capacidades militares conforme a la línea estratégica del Ministerio de Defensa.

- Punto de vista de adquisición (*Acquisition viewpoint*, AcV): describe detalles programáticos como dependencias entre actividades, líneas y márgenes temporales, etc.

Cada punto de vista produce varias vistas con diferentes niveles de detalle y para la especificación formal de las mismas MoDAF se apoya en los siguientes componentes: la propia definición de las vistas y cómo utilizarlas; el Metamodelo MoDAF (*MoDAF Meta Model*, M3) que describe los objetos arquitecturales disponibles y las relaciones entre ellos, definido como un perfil UML 2.0; la taxonomía MoDAF que proporciona un conjunto común y estructurado de nombres de objetos y definiciones; y el repositorio arquitectural del Ministerio de Defensa (*MoD Architectural Repository*, MoDAR) que almacena modelos arquitecturales. Para desarrollar las arquitecturas MoDAF cuenta con un proceso en seis pasos (al igual que DoDAF) además de varias variantes de este y otras alternativas. Además cuenta con un plan de certificación de herramientas software que asisten en el proceso de desarrollo de arquitecturas DoDAF y programas de formación a través de empresas externas.

3.5.8. Rational Unified Process (RUP) y Modelo de vistas "4+1" para arquitecturas software

RUP [44] es un proceso de ingeniería del software que describe flujos de trabajo y fases, con iteraciones para reducir riesgos en etapas tempranas del ciclo de desarrollo. Concibe cuatro fases en cada ciclo: origen (*inception*), elaboración (*elaboration*), construcción (*construction*) y transición (*transition*) (ver Apartado 5.1). Por otro lado, el conjunto de vistas de Kruchten "4+1" [45] organiza la descripción de arquitecturas software usando cinco vistas concurrentes, cada una centrando un conjunto específico de aspectos. Cuatro de las vistas asisten a la toma de decisiones de diseño y la quinta permite ilustrarlas y validarlas. Éstas son:

- Vista lógica (*logical view*): se corresponde con el punto de vista del usuario final y consta de diagramas de casos de uso, objetos y estados.

- Vista de desarrollo (*development view*): relacionada con los programadores y gestores cuenta con diagramas de subsistemas y módulos.

- Vista de proceso (*process view*): se centra en la funcionalidad del sistema, rendimiento, escalabilidad e interoperatividad.

- Vista física (*physical view*): se corresponde con la realización del producto final.

- Vista de validación: consta de escenarios que aúnan las cuatro vistas anteriores.

Estas dos iniciativas fueron combinadas buscando aportar a una metodología de desarrollo software madura y ampliamente utilizada como RUP un marco de especificación de arquitecturas a través de puntos de vista [46]. Aunque este esfuerzo fue bien recibido por la industria y la comunidad científica posee algunas debilidades originadas por la poca consistencia del modelo de vistas "4+1". En su concepción original los puntos de vista son brevemente descritos y sólo se profundiza en el contenido esperado por cada uno de ellos. El producto RUP proporciona más información sobre las vistas pero no llega a definirlas con el nivel de detalle necesario para evitar confusiones entre los arquitectos.

3.5.9. GCM - Generic Component Model

Esta iniciativa surge a principios de los años 90 como un marco arquitectural para el análisis y diseño de sistemas distribuidos de propósito general [47]. Entre sus características destacadas es su orientación a componentes en lugar de a objetos y la reutilización de los puntos de vista del modelo de referencia ODP para especificar su marco de trabajo. Los componentes (relacionados con la empresa, lógicos o tecnológicos) son caracterizados mediante diferentes niveles de abstracción y construyen la arquitectura a todos los niveles de granularidad pudiendo encontrase desde micro hasta macro-componentes. Este marco crea una matriz de especificación, como se puede ver en la Figura 2.10, a través de la conjunción de tres dimensiones:

1. Vector de granularidad (composición/descomposición de componentes). De mayor a menor nivel de composición tenemos: los conceptos de negocio, las redes de relaciones, las funciones y servicios básicos, y los conceptos básicos.

2. Vector de abstracción (vista de componentes). Utiliza los puntos de vista del modelo de referencia ODP (vista de empresa, de información, computacional, de ingeniería y de tecnología).

3. Reduce la complejidad del sistema mediante separación de dominios (por ejemplo médico, legal, financiero, administrativo, social, técnico, etc.).

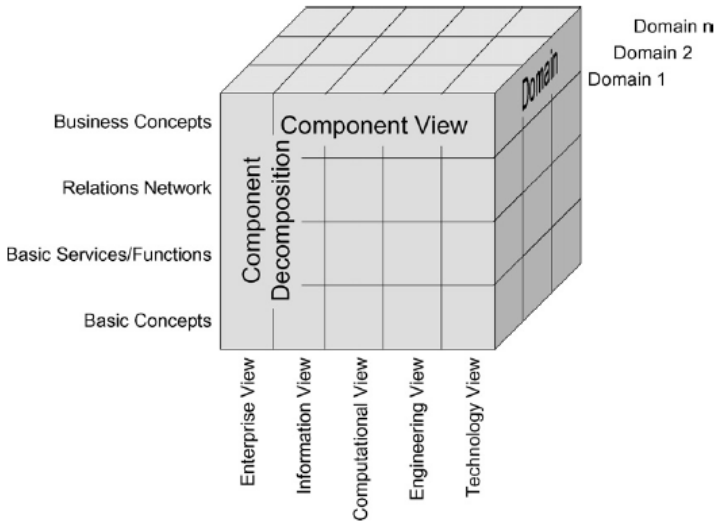

Figura 2.10. Dimensiones del marco de trabajo GCM

3.5.10. MEGAF

Trabajo publicado en el año 2010 [48] que pretende servir de infraestructura para la construcción de marcos de trabajo arquitecturales reutilizables y que hace las veces de metamodelo de marcos de trabajo. En su primera versión proporciona mecanismos para almacenar vistas, puntos de vista y aspectos del sistema y participantes; mecanismos para definir correspondencias entre vistas, puntos de vista, elementos arquitecturales, personas interesadas…; y mecanismos de comprobación de consistencia basada en relaciones y restricciones entre elementos.

3.5.11. Archimate

Esta iniciativa [49] es publicada en el año 2009 por parte de *The Open Group* (responsables de la metodología TOGAF) y consiste en un lenguaje de modelado de arquitecturas empresariales que permite describir los diferentes dominios arquitecturales y sus relaciones y dependencias subyacentes. Aporta un marco de trabajo pero en el sentido de un lenguaje común de descripción y no proviene de UML aunque herede algunas de sus características. A pesar de ser un trabajo de reciente aparición, las posibilidades de combinación con TOGAF han hecho que centre mucho interés y esfuerzos. Así podemos encontrar, por ejemplo, diversas herramientas software desarrolladas por empresas que soportan el modelado de arquitecturas a través del lenguaje Archimate.

El núcleo de conceptos del lenguaje se estructura en los grupos: elementos de estructura activa (comprende actores, componentes de aplicación, dispositivos, etc.; es decir, el conjunto de sujetos que realizan acciones), elementos de comportamiento (o servicios) y elementos de estructura pasiva (que agrupa los objetos que reciben las acciones). Archimate define tres niveles de especialización de los conceptos anteriores: negocio (*business,* relacionado con servicios orientados al usuario final), aplicación (*application,* servicios de aplicación realizados por componentes software) y tecnología

(*technology,* orientado a los servicios de infraestructura). Cada nivel se subdivide en los tres grupos descritos anteriormente y tenemos el marco de trabajo que se muestra en la Figura 2.11. Para cada nivel, Archimate define un metamodelo de conceptos que permite definir la arquitectura formalmente. Cabe destacar, por último, que las dependencias y relaciones entre niveles están muy bien definidas así como los mecanismos para especificarlas, lo que lleva a la construcción de modelos muy coherentes e integrados.

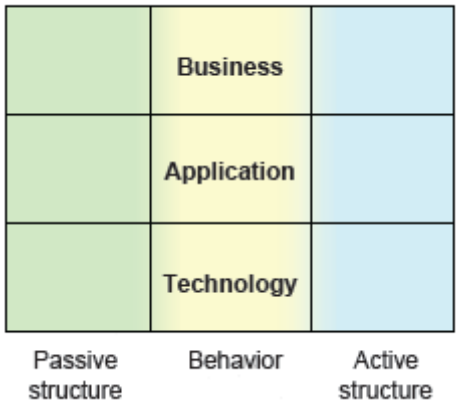

Figura 2.11. Niveles de especialización y grupos de conceptos de Archimate

En Archimate el término punto de vista se refiere al conjunto de conceptos que agrupados permiten especificar la arquitectura desde un aspecto concreto. Se definen dieciocho puntos de vista, esto es, agrupaciones de conceptos del lenguaje:

1. Vista introductoria (*Introductory view*): de alto nivel se centra en los procesos de negocio y consta de una notación simplificada para representar la arquitectura de forma sencilla.

2. Vista de organización (*Organization view*): permite especificar la organización interna de la empresa en términos de actores, recursos, localizaciones, departamentos, etc.

3. Vista de cooperación entre actores (*Actor co-operation view*): relaciones entre actores y de éstos con el entorno.

4. Vista funcional de negocio (*Business function view*): especifica las principales funciones de negocio de la organización y sus relaciones en cuanto a intercambio y uso de recursos, información, etc.

5. Vista de proceso de negocio (*Business process view*): estructura y composición a alto nivel de los procesos de negocio.

6. Vista de cooperación entre procesos de negocio (*Business process co-operation view*): relaciones entre procesos y de éstos con el entorno.

7. Vista de producto (*Product view*): conjunto de productos ofrecidos por la organización, procesos de desarrollo de productos, etc.

8. Vista de comportamiento de aplicación (*Application behaviour view*): permite especificar el comportamiento interno de una aplicación.

9. Vista de cooperación de aplicación (*Application co-operation view*): relaciones entre componentes de aplicación (flujos de información, recursos, etc.).

10. Vista de estructura de aplicación (*Application structure view*): define la estructura de una aplicación o componente.

11. Vista de uso de aplicación (*Application usage view*): permite especificar cómo se utilizan las aplicaciones para soportar los procesos de negocio.

12. Vista de infraestructura (*Infrastructure view*): elementos software y hardware de infraestructura que soportan la capa de aplicación.

13. Vista de uso de infraestructura (Infrastructure usage view): especifica cómo las aplicaciones son soportadas por la infraestructura.

14. Vista de implementación y despliegue (*Implementation and deployment view*): muestra cómo una o más aplicaciones son implementadas en la infraestructura.

15. Vista de estructura de información (*Information structure view*): permite definir los modelos de información utilizados.

16. Vista de realización de servicios (*Service realization view*): especifica cómo los servicios son realizados por procesos subyacentes.

17. Vista en niveles (*Layered view*): recoge varias capas y aspectos de una arquitectura de empresa en un único diagrama.

18. Vista de paisaje (*Landscape map view*): permite especificar las relaciones arquitecturales entre las diferentes capas.

Finalmente, la conjunción de esta iniciativa con TOGAF se basa en que Archimate proporciona un conjunto de conceptos independientes de proveedores y desarrolladores, incluyendo representaciones gráficas que permiten crear un modelo integrado y consistente de una arquitectura. Este modelo puede ser descrito a través de los puntos de vista de TOGAF. Además la estructura del lenguaje Archimate se corresponde con las tres arquitecturas de TOGAF (*Business, Information systems* y *Technology*) y esto facilita el mapeo entre algunas vistas TOGAF con los puntos de vista de Archimate. Esta correspondencia

no es completa ya que el alcance de TOGAF es más amplio que el del lenguaje Archimate, el cual no contempla aspectos estratégicos de alto nivel o de ingeniería de bajo nivel como hace TOGAF.

3.6. Resumen de estándares e iniciativas

En la Tabla 2.II se recoge información básica sobre el conjunto de estándares e iniciativas revisados en los apartados anteriores. Se destaca el año de aparición de cada estándar así como el de su última versión, el organismo que lo soporta y promueve, si tiene naturaleza abierta o cerrada y unos comentarios sobre el uso actual de cada uno.

Nombre	Año aparición	Año última versión	Organización promotora	Naturaleza	Uso actual
RM-ODP	1997	2009 (algunas partes)	ISO / ITU	Abierta	Industria y academia
MDA	2003	2005	OMG	Abierta	Industria y academia
TOGAF	1995	2009	The Open Group	Cerrada	Industria
Zachman Framework	1987	2008	Zachman Enterprise	Cerrada	Industria
SOMF	2008	2011	Methodologies Corporation	Cerrada	Industria
ARIS	1996	2006	IDS Scheer	Cerrada	Industria
Praxeme	2004	2010	Praxeme Institute	Abierta	Industria y academia
FEAF	1999	1999	Gobierno EEUU	Cerrada	Organismos gubernamentales
DoDAF	2003	2010	Dpto. Defensa EEUU	Cerrada	Organismos gubernamentales
MoDAF	2005	2010	Ministerio Defensa Reino Unido	Cerrada	Organismos gubernamentales
GCM	1991	2000	ninguna	Abierta	Academia
MEGAF	2010	2010	ninguna	Abierta	Academia
Archimate	2009	2009	The Open Group	Abierta	Industria

Tabla 2.II. Resumen de los estándares e iniciativas de formalización de arquitecturas

3.7. Armonización entre estándares

Como se ha podido comprobar, las iniciativas y estándares presentados en los apartados anteriores siguen el objetivo común de formalizar el diseño y desarrollo de sistemas distribuidos. En términos generales, un marco establece los términos y conceptos necesarios para llevar a cabo la descripción arquitectural de un sistema así como las relaciones entre conceptos y con términos que describen el mundo real. En esta categoría se encuentran los estándares e iniciativas RM-ODP, MDA, TOGAF, Zachman Framework, DoDAF, FEAF, Praxeme, el conjunto de puntos de vista "4+1", MEGAF, MoDAF, SOMF, GCM y ARIS. En general, todos estos marcos arquitecturales siguen las recomendaciones de la norma ISO 42010 y presentan una descomposición de los sistemas en vistas por lo que es posible establecer una armonización de los mismos estableciendo relaciones entre ellas. En la Figura 2.12 se ilustra la correspondencia entre el modelo de referencia ODP, la iniciativa del OMG MDA y los marcos arquitecturales TOGAF y Zachman.

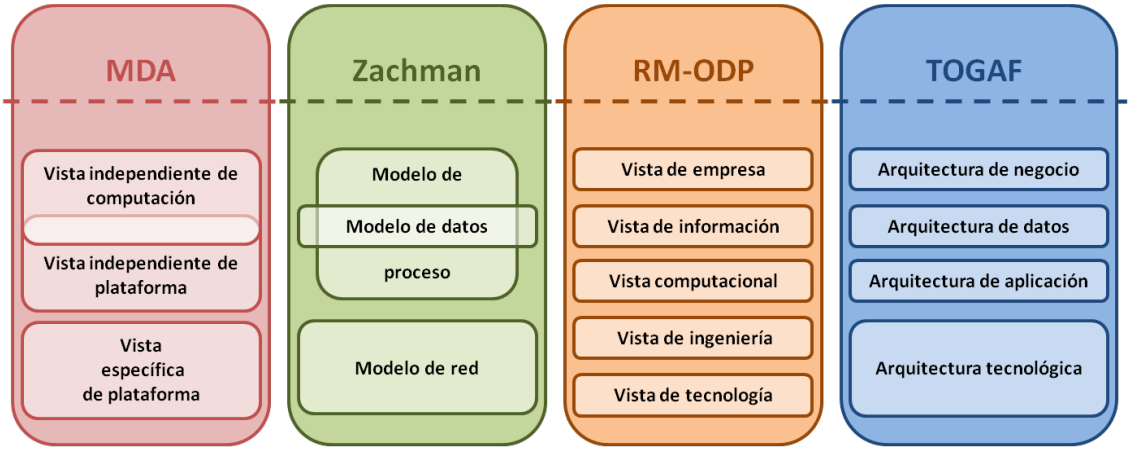

Figura 2.12. Relaciones entre vistas de los marcos de trabajo MDA, Zachman, RM-ODP y TOGAF

3.8. Lenguajes de formalización

Los marcos arquitecturales presentan guías, patrones y recomendaciones para la formalización de las arquitecturas pero son necesarias herramientas de modelado que faciliten el uso de un lenguaje, una notación y unos esquemas comunes que permitan formalizar los sistemas siguiendo las indicaciones de los estándares e iniciativas. A continuación se realiza una breve descripción de algunos de los más interesantes en este ámbito de trabajo.

3.8.1. UML - Unified Modeling Language

En general el análisis y diseño de sistemas en hardware y software está basado en modelos que describen el estado y/o comportamiento de los mismos así como la información que manejan. Uno de los lenguajes de modelado más extendidos es UML [19] el cual incluye una notación gráfica para crear modelos de sistemas software y aplicaciones. Su objetivo original es el de posibilitar la especificación, visualización, construcción y documentación de este tipo de desarrollos. La notación gráfica de UML permite representar conceptos comunes como clases, componentes, comportamientos, etc.; pero uno

de sus puntos fuertes es que su uso no está restringido al diseño de sistemas software, de hecho actualmente es utilizado también para el modelado de procesos de negocio, estructuras organizacionales, diseño de hardware, etc. A través de UML se pueden especificar distintas vistas de un modelo (en general cada vista está asociada a ciertos tipos de diagramas) comprendiendo:

- Vista de casos de uso: describen la funcionalidad del sistema percibida por actores externos.
- Vista de diseño: expresa los requisitos funcionales del sistema, entendiendo por ello los servicios que el sistema debería proporcionar a los usuarios finales.
- Vista de colaboración (contexto e interacción): posee una perspectiva interna del sistema para describir interacciones en el tiempo, en el espacio y concerniente al trabajo.
- Vista de escenario: describe un escenario mediante un camino de ejecución a través del sistema.
- Vista lógica: muestra cómo la funcionalidad está diseñada dentro del sistema en términos de estructura estática del sistema y comportamiento dinámico.
- Vista de componente: organización de los componentes software.
- Vista de procesos: cubre el comportamiento de los componentes, la capacidad de escalabilidad y rendimiento del sistema.
- Vista de concurrencia: muestra la concurrencia en el sistema destacando posibles problemas de comunicación y sincronización. Está muy centrada esta vista en la interoperatividad de componentes.
- Vista de implementación: comprende los componentes y archivos que se utilizan en la configuración física del sistema.
- Vista de despliegue: contiene los nodos que forman la topología de hardware sobre la que se ejecuta el sistema.

Las vistas presentadas utilizan un conjunto de diagramas del lenguaje UML para representar el sistema. Entre los diagramas más relevantes se pueden encontrar los siguientes:

- Diagrama de clases: de diseño estático, permite describir la estructura de un sistema mostrando sus clases, atributos, métodos y relaciones entre ellas.
- Diagrama de estados: cubre los aspectos dinámicos del sistema. Permite mostrar la historia de vida de un objeto, los eventos que causan la transición desde un estado a otro y las acciones que resultan debido a un cambio de estado. Los estados de una clase corresponden a todos los posibles estados de un objeto de su tipo. Cada estado de un objeto es una posible situación en la que el

objeto puede existir, durante la que realiza alguna actividad, espera algún evento o satisface alguna condición. Los estados se diferencian entre sí por los valores de los atributos y propiedades del objeto.

- Diagrama de componentes: cubre la vista de implementación estática de un sistema. Permite representar la separación de un sistema de software en componentes físicos (paquetes, componentes, archivos, etc.) y muestra las dependencias entre ellos.

- Diagrama de casos de uso: modelan aspectos dinámicos de un sistema a través de un conjunto de casos de uso, los actores implicados y sus relaciones.

- Diagramas de actividad: esencialmente son esquemas de flujo de actividad dentro del sistema. Modelan aspectos dinámicos del sistema.

Debido a la popularidad y usabilidad de UML se han propuesto diversas extensiones al lenguaje en la forma de perfiles. Un perfil en UML es un mecanismo genérico de extensión para adaptar los modelos UML a dominios y plataformas particulares. Los perfiles se definen utilizando estereotipos, etiquetas de definición y restricciones aplicadas a los elementos específicos de los modelos como Clase, Atributo, Operación, etc. Un perfil es una colección de tales extensiones que, colectivamente, particularizan UML para un dominio específico (por ejemplo, aeroespacial, sanitario, financiero) o plataforma (J2EE, .NET). A continuación se describen varios esfuerzos interesantes en el ámbito de este trabajo y que toman la forma de perfiles UML.

3.8.2. UML4ODP - UML para la especificación de sistemas ODP

En el año 2007 se publicó un estándar dentro de la familia del marco de referencia ODP orientado al modelado de sistemas [27]. Esta norma define un conjunto de perfiles UML para expresar la especificación de un sistema en términos de especificaciones de puntos de vista ODP. Utiliza conceptos definidos en UML y extensiones agrupados en un perfil UML 2.0 para cada punto de vista y una forma de utilizar dichos perfiles. También contempla el modelado de correspondencias ODP entre especificaciones de puntos de vista y un perfil para definirlas así como una vía de modelado de conformidad de implementaciones a especificaciones (puntos de referencia, enunciados de conformidad, etc.).

Esta norma viene a suplir el vacío que RM-ODP aquejaba en cuanto a notación. La flexibilidad de notación establecida a priori como un beneficio enfrentó duras críticas debido a que llevar los conceptos y relaciones del marco de referencia ODP a las herramientas de modelado era una tarea ardua y que ponía en riesgo a ODP como norma de referencia. Como respuesta a estas inquietudes surgió la norma UML4ODP facilitando la aplicación de RM-ODP a la especificación de sistemas.

Un aspecto clave es que esta norma potencia el beneficio de ODP al permitir formalizar la coherencia entre puntos de vista. Debido a que las vistas centran aspectos muy diferentes de los sistemas, la primera propuesta fue la de utilizar herramientas de modelado y convenciones diferentes para cada vista. El problema era que hacer corresponder los términos entre vistas y modelados a través de herramientas diferentes era complejo. La propuesta actual contempla que si la implementación de un sistema puede ser consistente con todas las vistas simultáneamente es porque existe consistencia entre ellas aunque ésta no se especifique explícitamente. En la norma UML4ODP se soluciona este aspecto haciendo que las correspondencias sean fragmentos de especificación separados, uno por cada par de vistas. Cada especificación de correspondencia se compone de un conjunto de enlaces de correspondencia (*correspondence links*) que pueden estar asociados a reglas estableciendo restricciones adicionales. Cada enlace tiene dos puntos finales que llevan etiquetas UML conteniendo un conjunto de términos en la vista apropiada. Se utilizan etiquetas que contienen la descripción textual de los términos, en lugar de enlazar los términos directamente, de esta forma se amplía la independencia entre vistas.

Otro aspecto que merece ser destacado es que esta norma permite el uso de herramientas de *Model-Driven Engineering* (MDE) para el desarrollo y mantenimiento de especificaciones de sistemas ODP, lo cual era otro punto criticado en RM-ODP. La norma UML4ODP representa los términos del RM-ODP pero también establece un metamodelado para cada uno de los puntos de vista lo que hace posible la aplicación de herramientas MDE para la transformación entre modelos.

3.8.3. SysML - Systems Modeling Language

SysML [50] nació de la necesidad de los ingenieros de sistemas de un lenguaje común para las aplicaciones en su ámbito de trabajo. De esta forma, el OMG colaborando con los equipos de trabajo de UML 2.0 acordaron en 2001 extender UML para que éste pudiera ser utilizado en las aplicaciones de ingeniería de sistemas. SysML personalizaría UML 2.0 a fin de soportar las especificaciones de análisis, diseño, verificación y validación de los sistemas complejos que pueden incluir componentes tanto hardware como software.

El principal objetivo de SysML es reutilizar un subconjunto de UML y proporciona extensiones adicionales para satisfacer los requisitos de la ingeniería de sistemas (Figura 2.13). Además se pretendía que SysML fuera simple pero a la vez una poderosa herramienta para el modelado de un amplio rango de problemas en ingeniería de sistemas, siendo particularmente efectivo para la especificación de requisitos, estructuras, conducta, asignación y limitaciones de los sistemas.

Por otro lado en el diseño de SysML se estudió el intercambio de datos entre modelos. En este sentido se tomaron dos posturas de forma que SysML soporta el intercambio de datos como lo hace UML, es decir, a través del OMG-XMI 2.0 [51]; y también con otras herramientas utilizadas en ingeniería de sistemas. El lenguaje SysML reutiliza y extiende muchos de los paquetes de UML 2.0. Las metaclases

UML se reutilizan y se unen dentro de un solo paquete de metamodelo, denominado UML4SysML. En la Figura 2.14 se recoge todo el contenido de este paquete así como las relaciones con las extensiones a UML especificadas por el perfil de SysML (*SysML Profile*).

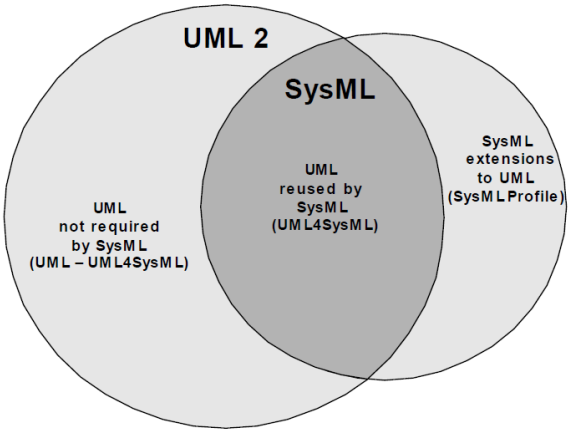

Figura 2.13. Relación entre UML y SysML

La semántica de los perfiles UML garantiza que, cuando un modelo de usuario "estrictamente" se aplica al perfil de SysML, sólo la metaclase UML referenciada por SysML está disponible para el usuario de ese modelo. Si el perfil no es "estrictamente" aplicado, metaclases UML a las que no se haga referencia explícita también pueden estar disponibles. Por otro lado el perfil de SysML también importa el perfil estándar L2 (*L2 Profile Standard*) de UML para hacer uso de sus estereotipos.

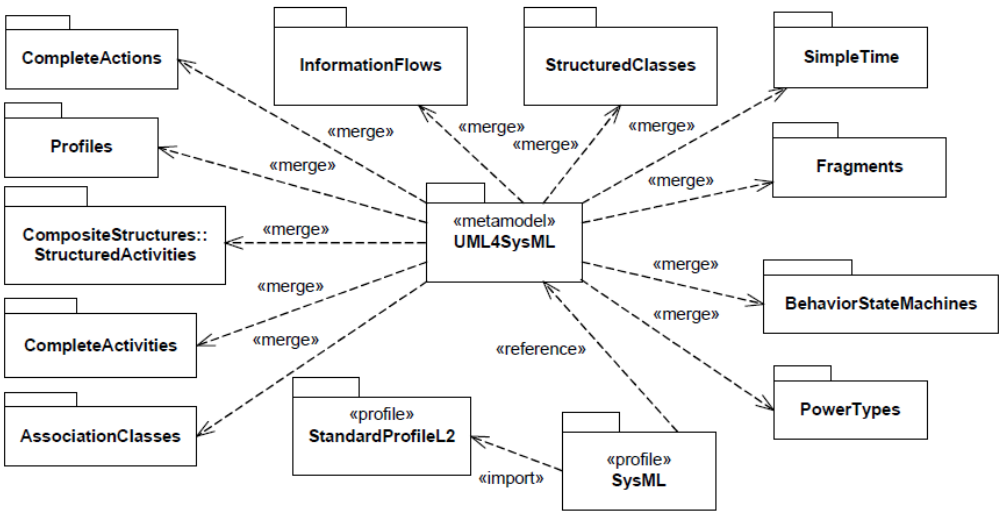

Figura 2.14. Herencias y extensiones entre UML y SysML

3.8.4. IDL - Interface Description Language

En general, un lenguaje de descripción de interfaz se utiliza para describir interfaces de componentes software de forma neutral, lo que permite la comunicación entre componentes desarrollados en diferentes lenguajes (por ejemplo, C++ y Java). Recordamos que una interfaz es un grupo de

operaciones que implementa un objeto, elemento clave para la encapsulación de los mismos ya que sólo a través de las interfaces pueden los objetos interaccionar entre sí. Una especificación IDL es un contrato entre un cliente y un servidor especificando la sintaxis de las interfaces. Una especificación en IDL comprende: declaraciones de módulos (permiten agrupar varias especificaciones IDL que presentan el mismo propósito), declaraciones de interfaces (soportando herencia y la especificación de operaciones y atributos) y declaraciones de tipos de datos, constantes y excepciones (necesarios para definir las operaciones y atributos). Aunque el IDL original data de 1983 [20], este mecanismo de especificación se ha extendido a otros propósitos y sistemas, existiendo actualmente numerosos IDL como el OMG IDL para CORBA o el WSDL para Servicios Web.

3.8.5. OWL-S - OWL para servicios

Otra herramienta de modelado, esta vez específica para la descripción de servicios semánticos, es OWL-S [12], una ontología construida sobre OWL y WSDL. A través de la formalización de las interfaces permite que los usuarios y agentes software descubran, invoquen, compongan y monitoricen de forma automática recursos semánticos que ofrecen servicios, bajo restricciones específicas. Esta ontología está compuesta de tres partes: el perfil de servicio (*service profile*) utilizado para describir lo que hacen los servicios. Esta información es utilizada por los usuarios e incluye el nombre y descripción de los servicios, limitaciones de uso y calidad de servicio, información del publicador y de contacto; el modelo de proceso (*process model*) describe cómo un cliente puede interactuar con el servicio. Esta descripción incluye el conjunto de entradas, salidas, precondiciones y resultados de la ejecución del servicio; y los fundamentos del servicio (*service grounding*) especifican los detalles que el cliente necesita para interactuar con el servicio como protocolos de comunicación, formatos de mensajes, números de puertos, etc.

4. Tecnologías de computación distribuida

4.1. Introducción

La historia y evolución de las tecnologías de computación distribuida comienza en los años 70. Al principio fue el modelo cliente-servidor el seguido para llevar a cabo las operaciones entre sistemas distribuidos. Este modelo es simple y eficaz para la organización de dichos sistemas pues los clientes generan peticiones y los servidores contestan. La comunicación es siempre iniciada por el cliente y los servidores pueden tener o no estado entre peticiones. Como primera implementación importante de este modelo surgen las Llamadas a Procedimientos Remotos (*Remote Procedure Call,* RPC) [52] que es el mecanismo básico de interacción en la mayor de parte del software de intermediación. Su objetivo es alcanzar la transparencia de localización y, para ello, utiliza representantes locales que codifican los datos entre sistemas. Una de las implementaciones más conocidas y extendidas del modelo RPC es la Invocación de Métodos Remotos (*Remote Method Invocation,* RMI) [53] de Java. Paralelamente a este desarrollo y complementándolo, a mediados de los años 80 se crea la representación externa de datos

(*External Data Representation*, *XDR*) [54], el primer esfuerzo importante en cuanto a estandarización en materia de sistemas distribuidos. La XDR es un formato estándar de serialización de datos que se utiliza en protocolos de redes de ordenadores y permitía que los datos fueran transferidos entre diferentes tipos de sistemas convirtiendo las representaciones locales al formato XDR portable entre sistemas operativos diferentes e independiente de la capa de transporte. Actualmente aún existe (en tres versiones diferentes) como estándar IETF.

Un esfuerzo propietario pero igualmente conocido de tecnología para comunicaciones distribuidas es el Modelo de Componentes de Objetos Distribuidos (*Distributed Object Component Model,* DCOM) [55] de Microsoft. El DCOM está compuesto por un intermediario de mensajes y un modelo de componentes para aplicaciones distribuidas de dicha empresa. Aunque es una solución mucho más compleja y directamente desplegable que RMI su fundamento es el mismo modelo básico RPC. DCOM fue un importante competidor de CORBA durante años aunque fue perdiendo fuerza con el tiempo.

4.2. CORBA – Common Object Request Broker Architecture

La arquitectura común para la intermediación en invocaciones a objetos (*Common Object Request Broker Architecture*, CORBA) [56] es un modelo de soporte para la programación de aplicaciones distribuidas y permite conectar clientes y servidores heterogéneos en los tres aspectos principales: sistema operativo, lenguaje de programación y hardware. El equipo tras el desarrollo de CORBA es el OMG que está compuesto por más de 800 proveedores, desarrolladores y usuarios de software, y cuya misión es promover la teoría y la práctica de la tecnología de objetos para el desarrollo de sistemas de computación distribuida. El objetivo es proporcionar un marco para una arquitectura común que soporte aplicaciones distribuidas orientadas a objetos y basadas en la reutilización de interfaces disponibles. Desde 1989, el OMG ha estado trabajando para crear estándares de componentes software basados en objetos en el marco de trabajo de su Arquitectura de Gestión de Objetos (*Object Management Architecture*, OMA) [57]. El elemento clave es CORBA cuya primera especificación fue adoptada en 1991. En 1994, en CORBA 2.0 se definió la interoperatividad entre objetos en sistemas heterogéneos. En CORBA 2.3 y 3.0 se han añadido nuevas funcionalidades, como por ejemplo el Adaptador de Objetos Portable, objetos por valor, el modelo de componentes CORBA, etc.

En la Figura 2.15 se ilustra la arquitectura propuesta por el OMG cuyos principales componentes son:

- El intermediador de invocaciones a objetos (*Object Request Broker*, ORB): también conocido como Bus de Objetos proporciona el mecanismo para que los objetos realicen peticiones y obtengan respuestas a estas peticiones de una forma transparente. El ORB simplifica la programación de objetos distribuidos ya que libera al cliente de implementar los mecanismos necesarios para las invocaciones de métodos remotos del servidor. Este elemento consigue que, desde el punto de vista del cliente, una llamada a un método remoto se realice de forma similar a la llamada a un

método local. El ORB no es un único elemento, sino que se compone de varias interfaces y cualquier implementación del ORB debe ajustarse a esas interfaces.

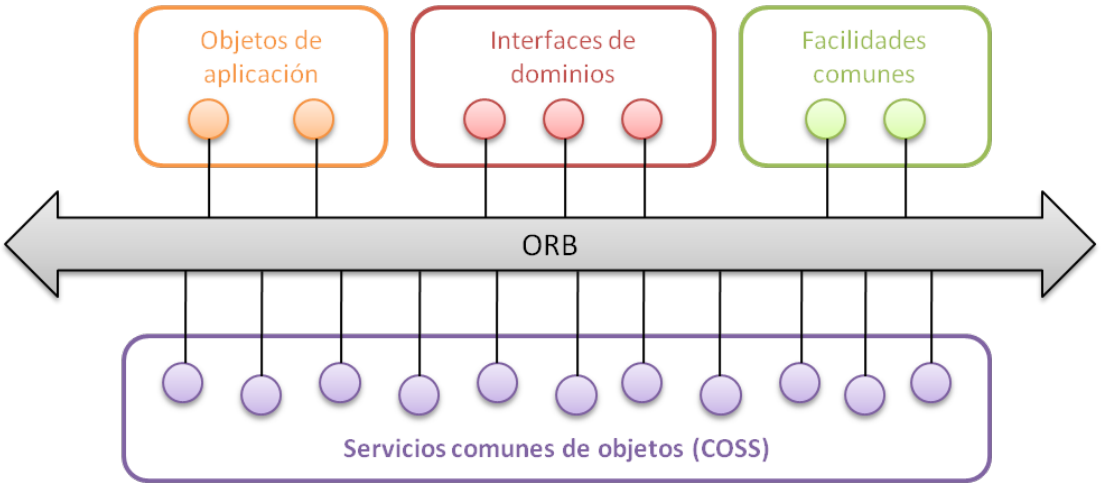

Figura 2.15. Arquitectura de gestión de objetos

- Especificaciones de Servicios Comunes de Objetos (*Common Object Services Specification*, COSS): ofrecen servicios fundamentales y de uso frecuente en diversas aplicaciones. Se encargan de aumentar y complementar la funcionalidad del ORB. Son considerados servicios de bajo nivel y entre ellos se encuentran el servicio de ciclo de vida, el de persistencia, el de transacciones, el de nombrado y el de eventos.

- Facilidades Comunes: ofrecen servicios (facilidades) de más alto nivel para el uso directo de los objetos de aplicación (pueden usar o especializar varios módulos COSS). Se configuran de acuerdo a la aplicación y son independientes del dominio de aplicación.

- Interfaces de Dominios: representan áreas verticales que proporcionan funcionalidades de interés para usuarios finales en dominios de aplicaciones particulares. Son dependientes del dominio de la aplicación (telecomunicaciones, fabricación, finanzas, etc.). La iniciativa CORBAmed (ahora *Healthcare Domain Task Force,* HDTF, ver Apartado 7.2) es un ejemplo de dominio, en este caso sanitario, que define sus propios objetos e interfaces de aplicación orientadas exclusivamente a su ámbito.

- Objetos de Aplicación: componentes específicos de las aplicaciones de usuarios finales. Al igual que el resto de objetos, éstos deben ser definidos usando IDL para poder participar en los intercambios mediados por el ORB. Una aplicación surge como resultado del ensamblado de estos objetos. Los Objetos de Aplicación están construidos encima de los servicios proporcionados por el ORB, de las Facilidades Comunes y de los Servicios Comunes.

Claramente se advierte que, para abordar la complejidad de la heterogeneidad, es necesaria la adopción de estándares que definan interfaces para que los objetos interoperen en entornos cliente/servidor

heterogéneos. Algo loable de OMG es que creó esos estándares antes que los objetos distribuidos se desarrollaran más. Así el lenguaje de definición de interfaz del OMG (*OMG Interface Definition Language*, OMG IDL) se utiliza para especificar los límites de los componentes y sus interfaces con los clientes. IDL es un lenguaje neutral y totalmente declarativo (no define detalles de implementación) que proporciona interfaces independientes de las aplicaciones o las tecnologías concretas de implementación para todos los servicios y componentes que residen en un bus CORBA. La definición IDL de un objeto CORBA contiene el conjunto de operaciones que un cliente puede invocar en este objeto, los tipos de datos que se intercambian y las excepciones que pueden ocurrir.

4.3. Servicios Web

El W3C [58] define los servicios web como sistemas software diseñados para soportar interacciones máquina a máquina sobre una red potenciando la interoperatividad [59]. Distintas aplicaciones software desarrolladas en lenguajes de programación diferentes y ejecutadas sobre cualquier plataforma pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperatividad se consigue mediante la adopción de estándares abiertos, así por ejemplo cada servicio web tiene una interfaz descrita en un formato procesable por las máquinas, concretamente WSDL, y los clientes interactúan con el servicio web como se prescribe en esa descripción utilizando mensajes SOAP [60], normalmente enviados usando mensajes XML del protocolo SOAP normalmente sobre HTTP [61]. Las tecnologías de servicios web se apoyan en un amplio conjunto de estándares y especificaciones como recoge la Figura 2.16.

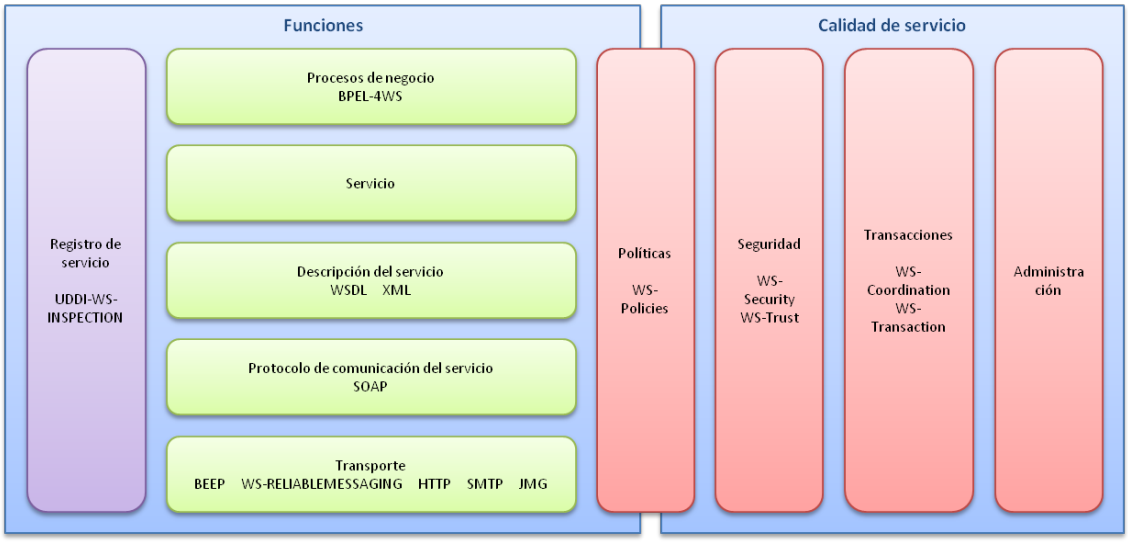

Figura 2.16. Arquitectura de los servicios web [59]

Esta arquitectura puede ser separada en dos generaciones asociadas a los estándares y especificaciones que utilizan. La primera generación contaba exclusivamente con las especificaciones del núcleo como WSDL, XSD (*XML Schema Definition Language*) [62], SOAP, UDDI (*Universal Description, Discovery, and Integration*) [63] y el Perfil Básico de integración WS-I [64]. Esta primera plataforma se utilizaba en

soluciones distribuidas tradicionales y proporcionaba canales de integración punto a punto pero presentaba enormes lagunas en cuanto a garantizar calidad de servicio, transacciones y comunicaciones fiables. Con la madurez y progresiva adopción de estos estándares, la plataforma de servicios web amplió el alcance de su utilización y aparecieron nuevas especificaciones, dando lugar a lo que se vino a llamar segunda generación. Estas extensiones proporcionan un conjunto de funcionalidades y aspectos más sofisticados tanto a la tecnología como al diseño de servicios web. Entre las ventajas que tienen los servicios web y que han contribuido a su amplia aceptación están su apoyo a la interoperatividad entre aplicaciones software independientemente de sus propiedades o de las plataformas sobre las que se despliegan, el fomento y uso de estándares y protocolos basados en texto que hacen más sencillo acceder a su contenido y entender su funcionamiento, el aprovechamiento de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado puesto que se apoyan en el protocolo HTTP, etc. Por otro lado cuentan también con algunos inconvenientes como que en la realización de transacciones no pueden compararse con el grado de desarrollo y sofisticación de estándares abiertos de computación distribuida como CORBA y el hecho de que al adoptar un formato basado en texto como XML su rendimiento es menor al de otros modelos de computación distribuida.

Actualmente, y aunque el modelo arquitectural orientado a servicios es independiente de tecnologías de implementación, la plataforma tecnológica más asociada a la realización de SOA son los servicios web. Esto se debe, entre otros aspectos, a la amplia aceptación de estos últimos en la industria y a que presentan un marco de comunicaciones basado en contratos/descripciones de servicios desacoplados de la implementación de los mismos que encaja adecuadamente con los principios SOA.

4.4. Procesamiento en Grid (*Grid computing*)

Mientras que los servicios web están orientados a compartir información a través, fundamentalmente, de Internet, la computación en Grid y sus servicios permiten además compartir potencia de procesado, capacidad de almacenamiento, y en general cualquier tipo de recurso, a través de la red. Esta compartición se consigue virtualizando los recursos que participan en una infraestructura grid de manera que para el usuario final actúan como un único y potente ordenador. Esta infraestructura (o software de intermediación) grid es el componente software esencial que permite la integración de los distintos tipos de recursos participantes y consiste en servicios grid, que comúnmente están basados en la tecnología de servicios web. En un sentido amplio, podemos considerar la Web como una “Grid de información” y la Grid como una “Web extendida” que permite a los usuarios compartir otros tipos de recursos más allá de información. A lo largo de los años han ido surgiendo muchas soluciones, cada una de ellas desarrollando sus propios elementos e implementando el concepto de Grid de forma diferente. No es hasta el año 2002 cuando el Proyecto Globus (una amplia comunidad de investigadores centrados en el procesamiento en Grid, ver Apartado 4.5) [65] e IBM [66] sientan las bases para una especificación de estos sistemas basándose en tecnologías ampliamente extendidas y maduras como los servicios web. Fue necesario añadir algunas características de las que carecían éstos (estado, transitoriedad, servicio de

notificaciones…) creando así lo que durante un tiempo se denominaron “servicios grid”. Las bases del procesamiento en Grid se encuadraron en la especificación OGSA (*Open Grid Services Architecture*) [67] la cual permite la integración de servicios y recursos a través de entornos y comunidades distribuidas, heterogéneas y dinámicas. Para alcanzar esta integración, el modelo OGSA adoptaba el estándar WSDL y definía el concepto de servicio grid sobre el de servicio web. En el año 2004 se publicó el estándar WSRF (*Web Service Resource Framework*) [68] que extendía los servicios web para incorporar las funcionalidades que aportaba un servicio grid.

Una de las claves del procesamiento en Grid es el concepto de organización virtual (*virtual organization*) [69] un esquema de distribución que incluye un conjunto de organizaciones y entidades colaborando entre sí compartiendo sus capacidades en forma de los servicios y recursos que poseen, todo ello buscando resolver problemas complejos dentro del entorno de red. En las relaciones que se establecen entre las entidades de las distintas organizaciones el uso compartido de recursos va más allá del simple intercambio de documentos, incluyendo acceso directo a software remoto, capacidad de procesamiento, datos, dispositivos sensores/actuadores y otros. El uso compartido de recursos que se lleva a cabo en las arquitecturas Grid tiene que ser altamente controlado. Los proveedores y consumidores de recursos deben definir claramente qué está siendo compartido, a quién se le está permitido el acceso, y las condiciones bajo las cuales ese acceso puede realizarse. Así la definición formal de organización virtual sería la de un conjunto de individuos y/o instituciones junto a las reglas que determinan y restringen la colaboración entre ellos.

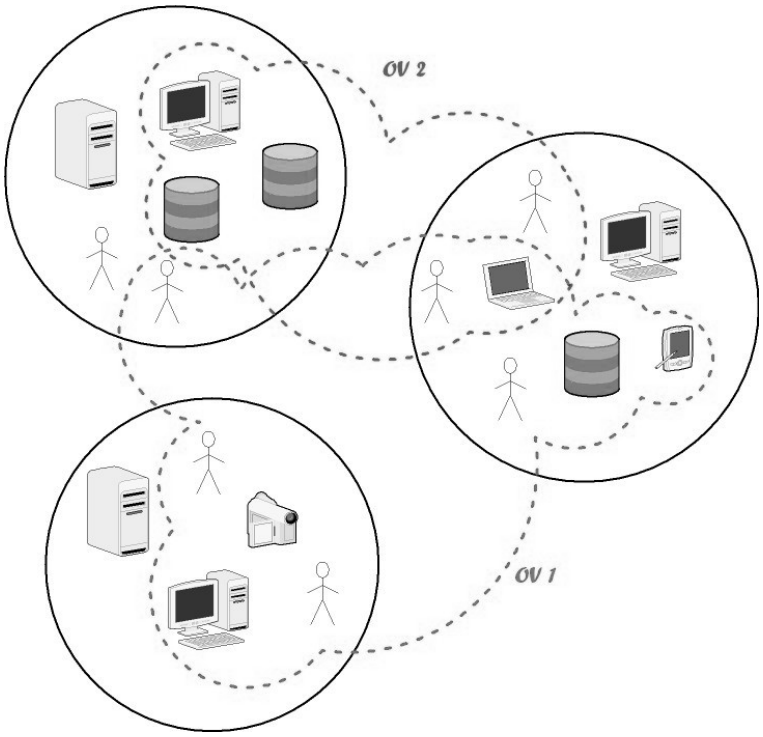

Figura 2.17. Ejemplo de organizaciones virtuales sobre organizaciones físicas

A partir de OGSA (e incluso algunas también al margen de ella) surgen las primeras implementaciones de software de intermediación que permiten desarrollar aplicaciones basadas en Grid (*The Globus Toolkit* [65], *gLite* [70], *Unicore* [71]…). Una de las más extendidas, más conforme a las especificaciones Grid y que mayor evolución ha sufrido es *The Globus Toolkit* que proporciona un conjunto de componentes y servicios que cubren gran parte de los aspectos involucrados en este procesamiento distribuido, por lo tanto puede ser utilizada como base para soluciones que exploten las bondades de la Grid.

4.4.1. OGSA – The Open Grid Services Architecture

El OGF (*Open Grid Forum*) [72] es una comunidad abierta que tiene como objetivo promover y dar soporte al desarrollo, despliegue e implementación de las tecnologías y aplicaciones grid mediante la creación de documentación consensuada, guías de mejores prácticas, especificaciones técnicas, guías de implementación, etc. Su principal aportación para la normalización de sistemas grid es el estándar OGSA que define un software de intermediación abierto que proporciona transparencias de distribución a los sistemas grid. Como se puede ver en la Figura 2.18, OGSA especifica tres capas:

1. Capa de acceso: proporciona los puntos de interacción entre los usuarios y la Grid ya sea mediante una aplicación construida con este propósito específico, una pasarela, un conjunto de APIs, etc.

2. Capa de servicios: soporta un conjunto de interfaces estándar que permite la conexión entre la capa de acceso y los recursos. Describir este nivel es el objetivo primordial de OGSA y el OGF.

3. Capa de recursos: proporciona los puntos de conexión y manipulación de los recursos ya sean virtuales o físicos.

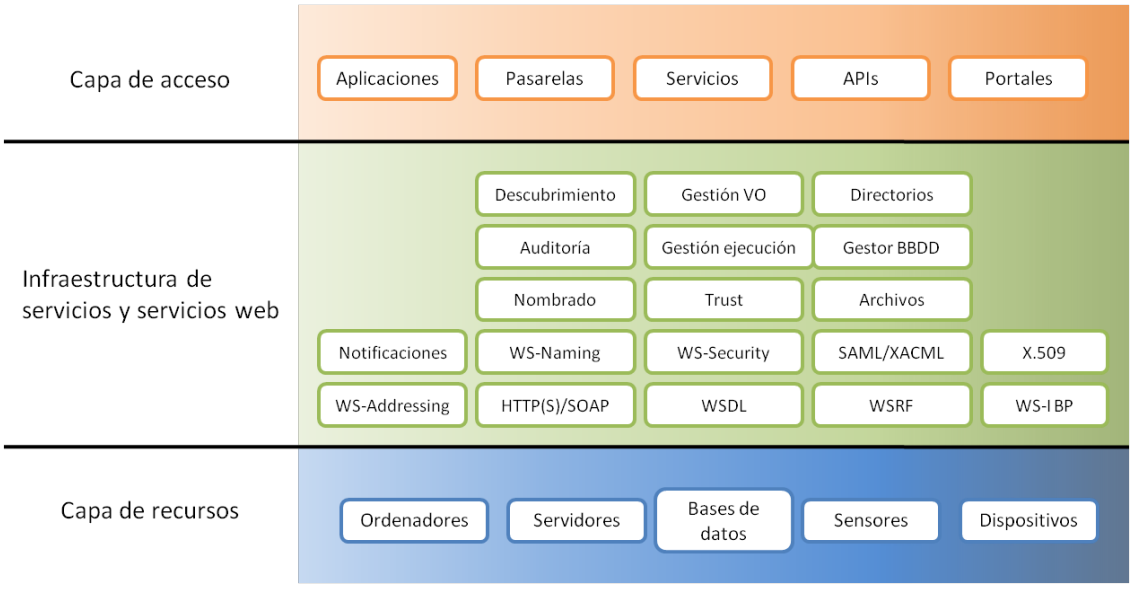

Figura 2.18. Arquitectura de la especificación OGSA [65]

Uno de los principales beneficios de este modelo en capas es que las interfaces de los servicios están completamente desacopladas de los recursos que utilizan o de la implementación subyacente, lo que permite que cualquier dispositivo que pueda conectarse a una red utilice las capacidades de los servicios OGSA.

La capa de servicios o infraestructura está compuesta de un conjunto de funcionalidades comunes y necesarias para la implementación de servicios distribuidos de alto nivel. Cuando OGSA está construida utilizando tecnologías de Servicios Web el lenguaje WSDL define las interfaces de servicio. Además la infraestructura incorpora estándares como WSRF, WS-Management [73] y WS-Addressing [74]. Los servicios de infraestructura resuelven aspectos de la distribución como el nombrado, la comunicación o la seguridad; aunque hay que destacar que OGSA es una especificación de software de intermediación para computación en Grid pero independiente de implementaciones tecnológicas, o lo que es lo mismo, los servicios están identificados pero no resueltos tecnológicamente. Esto será el objetivo de herramientas como *The Globus Toolkit*. La relación entre OGSA, WSRF y las tecnologías de Servicios Web puede verse en la Figura 2.19.

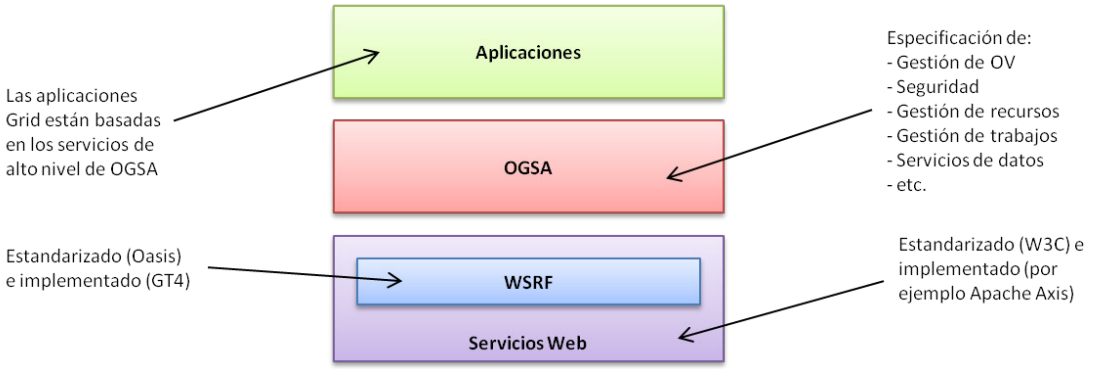

Figura 2.19. Relación entre OGSA, WSRF y Servicios Web

4.4.2. Servicios grid

Un servicio grid proporciona un conjunto de interfaces bien definidas y que siguen unas convenciones específicas y, más concretamente, es una (potencialmente transitoria) instancia de servicio con estado que soporta invocación fiable y segura (cuando se solicita), gestión del tiempo de vida, notificación, gestión de políticas, gestión de credenciales y virtualización. En el modelo OGSA, todo es representado como un servicio grid, ya sea un recurso computacional, de almacenamiento, una red, una aplicación, una base de datos, etc. La diferencia entre OGSA y los servicios web es que en la primera los servicios grid pueden ser gestionados (creados, monitorizados, destruidos, etc.). La arquitectura OGSA además permite a una aplicación obtener referencias a instancias de servicios grid que pueden ser usadas para monitorizar un servicio y acceder a sus datos locales a través de las interfaces definidas, en principio de forma similar a la funcionalidad proporcionada por los sistemas de objetos distribuidos. Sin embargo, todo esto se realiza a través de documentos XML que son intercambiados por los servicios grid, de la

misma forma que los servicios web. Los servicios grid, al igual que los web, son *servicios virtuales*, los cuales proporcionan una interfaz consistente para diversas implementaciones finales. El acceso a los recursos se hace de forma transparente, lo que permite el mapeo de múltiples instancias lógicas de recursos al mismo recurso físico. En el contexto de la Grid, los servicios ayudan a construir una capa virtual sobre los recursos grid que permite mapear el comportamiento semántico de servicios comunes en múltiples plataformas.

4.5. The Globus Toolkit

El conjunto de herramientas Globus [65] es un esfuerzo de código abierto cuyo objetivo es facilitar la construcción y despliegue de sistemas Grid. El paquete Globus incluye servicios y librerías software para la monitorización, descubrimiento y gestión de recursos así como gestión de archivos y seguridad. En parte debido a su estrategia de código abierto ha recibido enorme apoyo desde su versión inicial en 1998 (recientemente se publicó la quinta versión) por una comunidad abierta de desarrolladores y está siendo utilizado en numerosos proyectos de ciencia e ingeniería mientras que es el germen de productos comerciales Grid. El paquete Globus contiene un conjunto de componentes (Figura 2.20) que pueden ser utilizados de forma independiente o combinados para desarrollar aplicaciones.

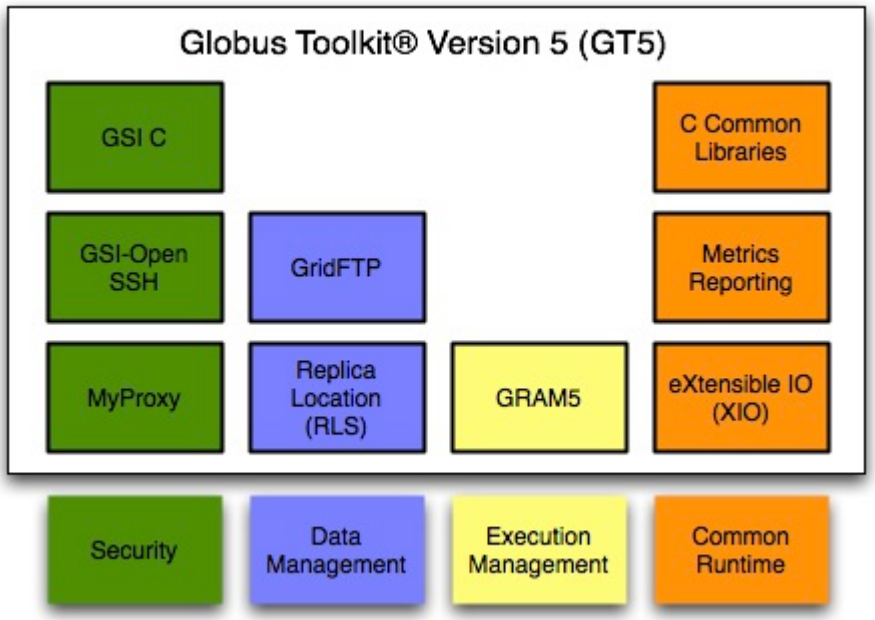

Figura 2.20. Áreas y componentes que conforman el conjunto de herramientas Globus

Esta herramienta fue concebida para eliminar los obstáculos de la colaboración entre organizaciones siguiendo así con el paradigma de organizaciones virtuales soportadas por la tecnología de computación en Grid. Sus servicios, interfaces y protocolos permiten a los usuarios acceder a recursos remotos ya estén dentro de su organización o en otra. Desde su segunda versión adoptó los estándares de servicios Grid como OGSA y WSRF convirtiéndose en el software de intermediación *de facto* y abierto para la construcción de Grids. Una muestra de la amplia difusión de este conjunto de herramientas es el

desarrollo de tecnologías basadas en Globus a través de los proyectos *European Data Grid* [75], *Grid Physics Network* (GriPhyN) [76], *Particle Physics Data Grid* [77] y otros.

4.6. DDS - Data Distribution Service for Real-time Systems

El DDS [78] es una especificación adoptada por el OMG para el intercambio de datos en tiempo real en sistemas distribuidos que fue creada en respuesta a las necesidades de la industria de estandarizar los sistemas centrados en datos. Proporciona una gran variedad de políticas de servicio que permiten una configuración muy precisa de la comunicación entre elementos participantes. De este modo se posibilita que DDS soporte varios tipos de requisitos temporales para sistemas en tiempo real.

La comunicación en DDS sigue el patrón publicador-suscriptor. Conecta por un lado a los productores de datos o publicadores y por otro a los consumidores de datos o suscriptores, desacoplándolos en tiempo y espacio. De esta manera, cada elemento no necesita conocer el origen de la información, ni cuándo ha sido producida, ni cómo llega hasta ella, sino únicamente basta con precisar qué tipo de información es la que se quiere y unas condiciones de comunicación que vienen dadas por las políticas de calidad de servicio (QoS). El contenedor de la información y nexo de unión entre los participantes de un dominio (publicadores y suscriptores) es el *topic* que lleva asociado un nombre y un tipo especificado en IDL (*Interface Definition Language*). Por otro lado, varios topics pueden agruparse formando un *multitopic* permitiendo la publicación/suscripción dentro de contextos más amplios. En la Figura 2.21 se muestra un escenario de ejemplo.

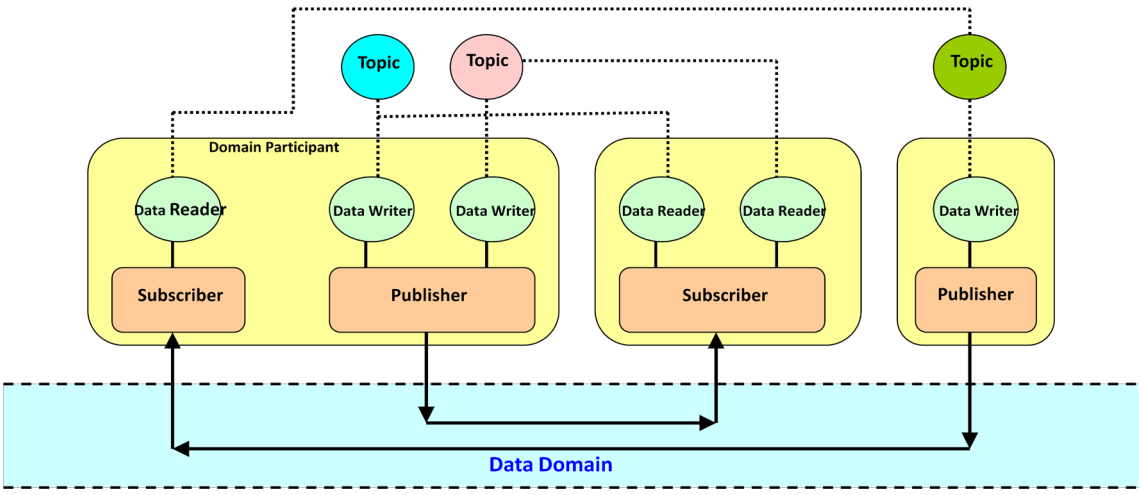

Figura 2.21. Escenario de ejemplo del patrón publicador-suscriptor

4.7. La integración semántica

Las herramientas de descripción y procesamiento de semántica aparecieron en la década de los 90 sentando las primeras bases para la interoperatividad semántica. Los medios de comunicación y el ámbito de intercambio de datos habían proliferado enormemente y se hacían necesarias tecnologías que permitiesen comprender la información intercambiada. El primer paso en esta disciplina fue la

conceptualización del mundo a través de ontologías, es decir, la descripción del ámbito en el que se desarrollaban los contenidos de los mensajes. Una conceptualización es una vista abstracta y simplificada del mundo que se desea representar para algún propósito. Cada base de conocimiento, sistema basado en conocimiento o agente de nivel de conocimiento está comprometido con alguna conceptualización, ya sea explícita o implícita. Una ontología es una especificación explícita de una conceptualización, en concreto define los términos y relaciones básicas que componen el vocabulario de un dominio así como las reglas que permiten combinar dichos términos y relaciones para definir extensiones del vocabulario [79]. Concretando para el dominio tecnológico, una ontología consiste en un conjunto de primitivas representacionales con las cuales se modela un dominio de conocimiento o discurso [80]. Estas primitivas son típicamente clases o conceptos (representando cualquier entidad sobre la que se puede decir algo, a la que se le puede poner nombre), atributos (o propiedades) y relaciones (o interacciones entre conceptos del dominio). Las definiciones de estas primitivas incluyen información sobre su significado y restricciones en su aplicación lógicamente consistente con el resto de primitivas. Algunos de los lenguajes lógicos de especificación de ontologías más extendidos son RDF [81], OWL [13] y DAML [82].

Las ontologías permiten compartir el entendimiento común de la estructura de información entre personas y agentes software. Por ejemplo si varios servicios web comparten y publican la misma ontología subyacente de los términos que utilizan, entonces agentes software pueden extraer y agregar información de esos servicios y servir como entrada a otras aplicaciones. Además las ontologías facilitan la reutilización del conocimiento de un dominio. Una ontología conceptualizando un dominio puede ser reutilizada en el mismo dominio por otros servicios o aplicaciones. Además, si se necesita construir una ontología que cubra todo un dominio, es posible integrar varias ontologías existentes que describen porciones del dominio completo. También se pueden reutilizar ontologías generales y especializarlas para describir dominios de interés específicos. Finalmente, las ontologías permiten analizar el conocimiento de un dominio y exponer características y relaciones implícitas al mismo.

Las ontologías, como conjuntos de términos y semánticas que son, pueden ser consideradas artefactos pasivos y por tanto su explotación y uso se debe hacer a través de aplicaciones. Una de las principales herramientas que permiten extraer grandes beneficios de las ontologías son los motores de inferencia. Estos son programas software que, a partir del conocimiento descrito explícitamente en una ontología, permiten derivar conocimiento implícito, obtener conclusiones y responder preguntas sobre el dominio. Existen numerosos motores de inferencia (como Jess [83] o Pellet [84]) pero éstos son sólo una muestra del amplio abanico de herramientas semánticas que se pueden encontrar actualmente en desarrollo y/o aplicación. Así podemos encontrar lenguajes de reglas sobre ontologías como SWRL (*Semantic Web Rule Language*) [85], lenguajes de consulta como SPARQL (*SPARQL Protocol and RDF Query Language*) [86] o entornos de trabajo como Protégé [87].

4.8. Otras tecnologías de integración

Terminamos este apartado presentando brevemente otras tecnologías que existen actualmente para la integración de componentes distribuidos y heterogéneos.

4.8.1. EBS - Bus de Servicios de Empresa (*Enterprise Service Bus*)

Un EBS [88] es otra propuesta de arquitectura software que proporciona una capa de abstracción entre productores y consumidores de mensajes permitiendo integrar diferentes tecnologías y sistemas. Propone un diseño de comunicación asíncrona orientada a mensajes entre aplicaciones. Las principales actividades de un ESB son monitorizar y controlar el encaminamiento de mensajes entre servicios, resolver conflictos entre componentes de servicios en comunicación, controlar el despliegue y versiones de los servicios, gestionar el uso de servicios redundantes, y proveer de servicios comunes como los de gestión de eventos o de coreografía, de transformación y mapeo de datos, de encolado y secuenciación de mensajes y eventos, de gestión de excepciones o seguridad, etc.

4.8.2. JMS - Java Message Service

JMS [89] es la solución creada por Sun Microsystems para el uso de colas de mensajes en sus aplicaciones Java. Es un estándar de mensajería que permite a los componentes de aplicaciones crear, enviar, recibir y leer mensajes. También hace posible la comunicación confiable de manera síncrona y asíncrona. Esta API permite implementar dos modelos diferentes:

- modelo punto a punto: que cuenta sólo con dos clientes, uno que envía el mensaje y otro que lo recibe. Este modelo asegura la llegada del mensaje ya que si el receptor no estuviera disponible para aceptar el mensaje o atenderlo, de cualquier forma se le envía el mensaje y éste se encola en una pila de tipo FIFO para luego ser recibido según el orden de llegada.

- modelo publicador/suscriptor: cuenta con varios clientes, los que publican *topics* o eventos y los que ver esos *topics.* A diferencia del modelo punto a punto en éste se tiende a tener más de un consumidor.

4.8.3. Plataformas de agentes

Con origen en el año 1996 la FIPA (*Foundation for Intelligent Physical Agents*) [90] es una organización dedicada a promover las tecnologías de agentes basadas en estándares. Desde el año 2005 pertenece al comité de estandarización del IEEE [91]. Las especificaciones FIPA abordan las diversas características que deben cumplir las plataformas de gestión de sistemas multi-agente y son ampliamente aceptadas y conformadas por los entornos de desarrollo y ejecución de agentes. Las especificaciones definen solamente el comportamiento de la plataforma de agentes a modo de interfaces de interacción. Entre los aspectos que considera la arquitectura están la creación/destrucción, el registro y localización, y la comunicación entre agentes. Además se definen un conjunto de servicios básicos que debe proporcionar la plataforma de agentes como el transporte de mensajes internos de la plataforma,

sistema de gestión de agentes, servicio de directorio y canal de comunicación de agentes. La comunicación entre estos servicios se hace a través de mensajes ACL (*Agent Communication Language*) [92] además de una ontología definida de conceptos. Por otro lado, la seguridad está basada en estándares como TSL [93] y Certificados X.509 [94]. Las ontologías se utilizan para establecer un lenguaje común entre agentes lo cual es necesario debido a que los agentes pueden ser de muy diversa naturaleza y origen. Cuando un agente es creado, se registra en el servicio de directorio e informa de qué ontologías tiene en su conocimiento. Un agente que quiera comunicarse con este tendrá que conocer alguna de las ontologías pues si no la comunicación a nivel semántico es imposible. Los sistemas multi-agente complejos mantienen unos agentes de ontología (*Ontology Agents*) que almacenan ontologías de uso público, marcan relaciones entre ontologías y pueden responder consultas.

5. Metodologías y marcos arquitecturales en el dominio sanitario

Como se ha visto, el campo de formalización de arquitecturas y sistemas distribuidos ha centrado numerosos esfuerzos desde la década de los 80. Actualmente se pueden encontrar diversas propuestas (estandarizadas o no) capaces de cubrir la especificación de todo tipo de sistemas, independientemente de su alcance, propósito o contexto. Debido al cambio organizacional que está teniendo lugar en el dominio sanitario y que aboga por una distribución y sofisticación de los sistemas, la aplicación de metodologías y marcos arquitecturales de formalización de arquitecturas se está generalizando. Tanto es así que han surgido en los últimos años diversas iniciativas que o bien particularizan marcos arquitecturales de propósito general al dominio sanitario (centrándose en los requisitos especiales que presenta este ámbito) o introducen nuevos métodos de formalización de arquitecturas exclusivos para el campo de la sanidad. En este apartado revisamos las iniciativas más relevantes.

5.1. HISA - Health Informatics - Service Architecture

Debido a la amplia aceptación que tuvo el modelo de referencia ODP desde su publicación, otros esfuerzos heredaron sus fundamentos con el objetivo de formalizar arquitecturas de sistemas distribuidos en dominios de aplicación concretos. En el año 2007 el comité de estandarización europeo (CEN) publicó el estándar HISA [2] cuyo foco estaba en identificar, por un lado, una metodología para describir sistemas de información sanitarios a través de un lenguaje, notación y paradigmas adecuados que permitan planear, diseñar y comparar sistemas; y, por otro, los aspectos arquitecturales fundamentales que facilitan la apertura, integración e interoperatividad de sistemas de información sanitarios. En el año 2009 la organización ISO adoptó el estándar HISA elevándolo a categoría de norma internacional.

El estándar HISA define una arquitectura para sistemas de información sanitaria de acuerdo al marco de referencia ODP, esto es, estructurada en vistas. Buscando mantenerse lo más general posible, en HISA sólo se ofrecen aspectos generales de la arquitectura sobre las tres vistas independientes de tecnología (puntos de vista de empresa, de información y computacional) mientras que las dos vistas dependientes

de tecnología restantes deberán ser abordadas por el arquitecto. Por tanto, la norma se divide en tres partes:

- Parte 1 – Punto de vista de la empresa (*Enterprise viewpoint*): especifica las características generales de la arquitectura, formaliza la metodología de especificación y los criterios de conformidad. Además detalla el punto de vista de empresa de la arquitectura, especificando los flujos de trabajo y actividades fundamentales de los grupos de usuarios que deben ser soportados por el software de intermediación así como los requisitos generales que presentan.

- Parte 2 – Punto de vista de la información (*Information viewpoint*): describe el punto de vista de información de la arquitectura a través de sus características fundamentales (objetos de información, relaciones entre ellos, etc.).

- Parte 3 – Punto de vista computacional (*Computational viewpoint*): detalla el punto de vista computacional de la arquitectura proporcionando las bases que aseguran la interoperatividad entre diferentes especificaciones de ingeniería y tecnología. Así identifica modelos de objetos computacionales e interfaces aunque no pretende ser una especificación completa de todas las posibles interfaces necesarias para implementar un sistema de información sanitario.

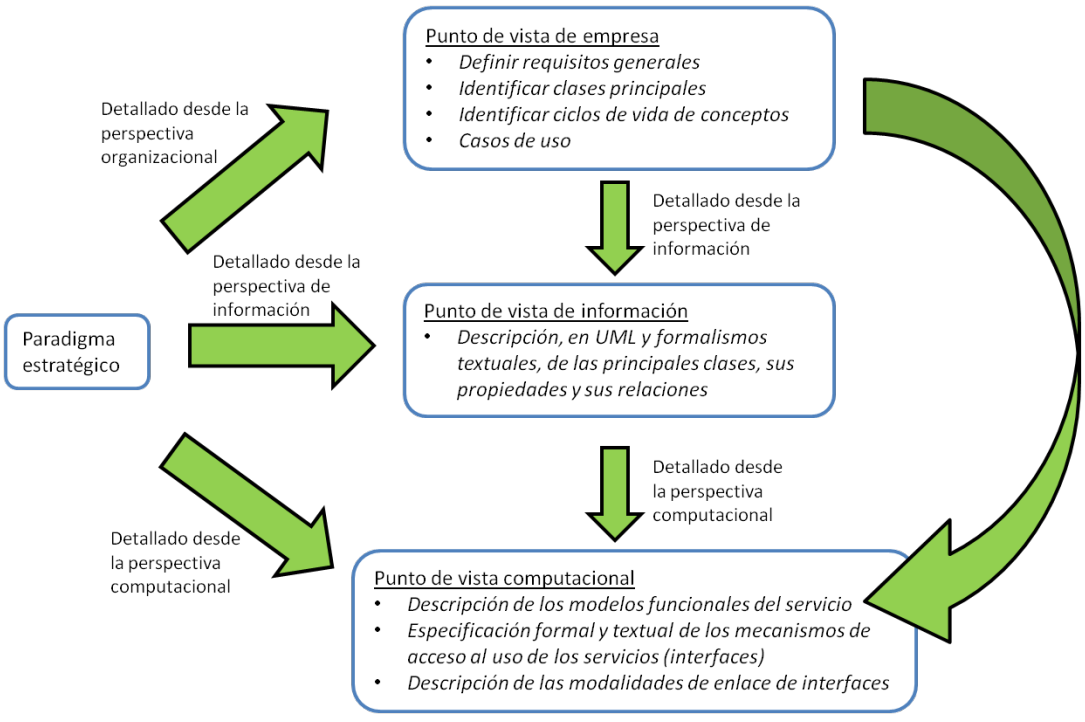

Figura 2.22. Los tres puntos de vista independientes de tecnología de RM-ODP detallados en HISA

HISA se basa en los puntos de vista y los lenguajes correspondientes del modelo de referencia ODP y establece una metodología para la especificación de la arquitectura aun cuando no estaba indicada en esa norma. El procedimiento de especificación HISA comienza con un documento, el Paradigma Estratégico (*Strategic Paradigm*), que identifica los requisitos generales y objetivos estratégicos del

sistema usando lenguaje natural para su descripción. A partir de él se especificarán las vistas de empresa, información y computacional utilizando los mecanismos provistos por el RM-ODP (Figura 2.22). Dependiendo de la complejidad del sistema, el proceso de especificación puede ser llevado a cabo de forma iterativa detallando cada vista a través de múltiples y secuenciales niveles de refinamiento. Al menos deberán ser proporcionados dos niveles: el estratégico (orientado a propósitos de decisión y planificación) y el operacional (para llevar a cabo las tareas de diseño y comparación). La especificación a nivel operacional puede realimentar a la especificación a nivel estratégico, dando lugar a modificaciones que a su vez repercutan en refinamientos del nivel operacional (Figura 2.23).

La norma HISA tiene poco tiempo de vida y no es un estándar plenamente maduro. De hecho, aunque cumple el objetivo para el que fue propuesta (definir aspectos esenciales de una arquitectura de sistemas de información sanitarios), no se ha conseguido alzar como el estándar de referencia para la formalización de arquitecturas en el ámbito sanitario como se esperaba. Entre los principales puntos que han sido criticados se pueden encontrar los siguientes:

1. Desde su concepción, el estándar HISA pretende ser un marco de trabajo abierto y no un conjunto completo y final de especificaciones. Así formaliza sólo aspectos concretos identificados como comunes y esenciales en cualquier sistema sanitario de información avanzado. Esta flexibilidad tiene como consecuencia que ciertas áreas no sean cubiertas en detalle como por ejemplo los fundamentos de seguridad tan esenciales en los sistemas de información sanitarios [95].

2. Por otro lado, aunque HISA hereda del marco de referencia ODP, dista mucho de especificar los tres puntos de vista de su arquitectura conforme a este estándar. El único punto que utiliza de RM-ODP es la separación en puntos de vista pero sin embargo no se ajusta a la terminología ni a las guías de modelado de esta norma.

Figura 2.23. Proceso iterativo de especificación incremental

5.2. CHF - Connected Health Framework

En el año 2006 surge una iniciativa promovida por la compañía Microsoft con el objetivo de aunar y razonar consejos y guías para el diseño y construcción de sistemas sanitarios y sociales centrados en el paciente [96]. En el año 2009 se publica una segunda versión con extensiones en el mantenimiento de la salud de la persona e incluyendo los puntos de vista de otros actores sanitarios como son los familiares y cuidadores. Esta propuesta está basada en estándares para la integración de sistemas y se fundamenta

en cuatro conceptos arquitecturales clave: orientación a servicios, federación de datos, federación de seguridad y fiabilidad. El propósito último del marco de trabajo CHF es el de guiar en el proceso de construcción, mantenimiento y diseminación del "registro de vida" de la salud y asistencia social de una persona. Este registro, que condensa todos los datos relacionados con la salud y la asistencia social de una persona a lo largo de su vida, tiene que estar perfectamente protegido y es propiedad del sujeto, el cual dispone de mecanismos para controlar su uso y distribución.

El CHF se materializa en un conjunto de patrones de negocio (Marco de negocio o *Business framework*) para la integración de aplicaciones a nivel de negocio y una arquitectura de referencia (Marco técnico o *Technical framework*) buscando resolver la interoperatividad técnica. Ambos elementos pertenecen a puntos de vista distintos del mismo sistema pero están relacionados entre sí (ya que los patrones de negocio se apoyan en las capacidades ofrecidas por la arquitectura de referencia) y son independientes de plataforma favoreciendo su reutilización y portabilidad. El marco de negocio tiene como objetivos potenciar la agilidad y flexibilidad de las aplicaciones (a través de la reusabilidad, orquestación…), facilitar la integración de sistemas legados, mejorar la privacidad y confidencialidad, aplicar inteligencia de negocio a la asistencia sanitaria y social, fomentar las aplicaciones de apoyo a la decisión, etc. Este elemento se compone de patrones de negocio los cuales se definen como propuestas reutilizables para la solución de un problema de negocio particular, normalmente centrado en un proceso de negocio, y con garantías de éxito basadas en aplicaciones y experiencias previas. Posee cerca de doscientas funciones categorizadas en grupos funcionales:

- Centrado en el paciente/cliente (*Patient/Client functional group*)
- Centrado en los proveedores de asistencia (*Care Providers functional group*)
- Centrado en las actividades de asistencia (*Care activities functional group*)
- Centrado en los profesionales sanitarios (*Care professionals functional group*)
- Centrado en gestión de datos, métodos y estándares (*Standards, methods and data management functional group*)

Por otro lado, el CHF cuenta con la arquitectura de referencia cuyos objetivos son facilitar la fiabilidad del sistema, la gestión de identidad, autenticación y autorización, la disponibilidad de los datos y auditorías, escalabilidad, rendimiento, movilidad, etc. Esta arquitectura proporciona un conjunto integrado de servicios genéricos basados en estándares, escenarios y casos de uso. Entre éstos se encuentran los servicios de gestión de identidad (*Identity Management services*), los de autenticación y autorización (*Authentication and Authorization services*), los de descubrimiento y publicación de servicios (*Service Publication and Discovery services*), los de historia clínica electrónica (*EHR services*) y otros.

5.3. HIS-DF - Health Information System – Development Framework

HIS-DF [97] es una iniciativa publicada por primera vez en el año 2000 que hereda la filosofía del GCM y busca armonizar las distintas propuestas arquitecturales (MDA, RM-ODP, RUP, *HL7 Development Framework* [98]…) en un marco y una metodología de desarrollo de arquitecturas para sistemas y componentes de información sanitarios semánticamente interoperables. Esta propuesta soporta el diseño de modelos UML de análisis de negocio, de requisitos y de diseño, cada uno de distinta naturaleza y alcance. Al modelo de análisis de dominio (*Domain Analysis Model*, DAM) se le aplican estándares de información (por ejemplo, el modelo de información HL7) y se obtiene el modelo DAM armonizado. Esto mejora la calidad semántica del DAM creando anotaciones semánticas utilizando los modelos de información y vocabulario HL7. El núcleo del proceso HIS-DF sigue cinco pasos como puede verse en la Figura 2.24.

1. Elección del sistema a analizar (restringido a aquellos centrados en la información sanitaria).

2. Separación del sistema en el dominio de interés (ámbito sanitario) del resto de dominios.

3. Analizar la complejidad estructural en base a los cuatro niveles de composición del GCM intentando definir el conocimiento del dominio para permitir la integración semántica. En este paso se hereda de diversas fuentes como: el modelo de información de referencia HL7 (*Reference Information Model*, RIM) [99], los arquetipos del OpenEHR [100], y estándares de arquitecturas de sistemas de información sanitaria como HISA y OMG HDTF (*OMG Health Domain Task Force*) [101].

4. A continuación se aplica la descomposición en vistas de RM-ODP pero sólo especificando aquellas vistas independientes de tecnología (el resto se obtendrán más adelante a través de transformaciones MDA).

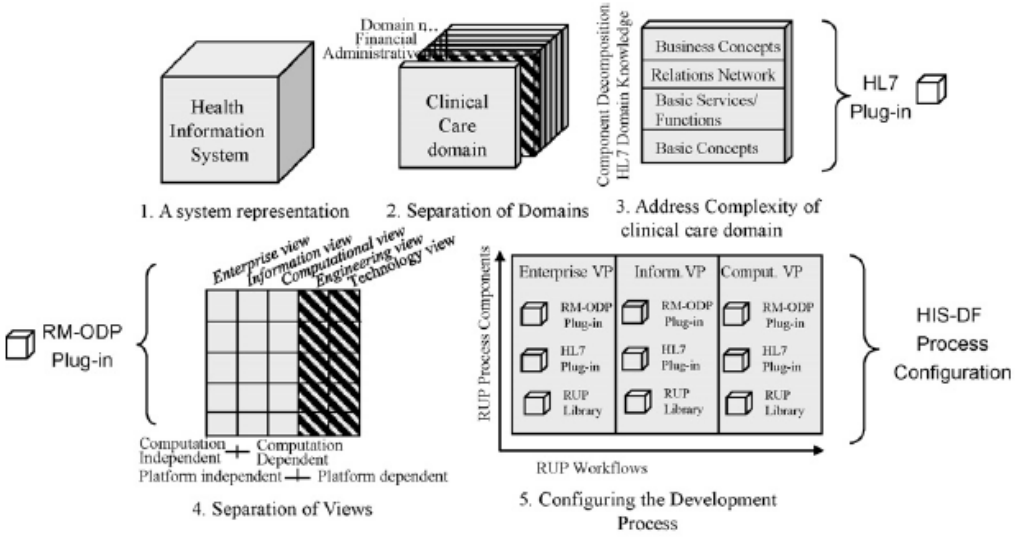

Figura 2.24. Proceso de desarrollo de HIS- DF

5. Por último se define un conjunto de componentes de proceso basado en el RUP (roles, productos de trabajo, artefactos, fases, guías, iteraciones y flujos de trabajo) para soportar cada punto de vista de RM-ODP ya que este estándar no propone una metodología o proceso de desarrollo en fases.

6. Software de intermediación y especificación de servicios en el dominio sanitario

Al igual que ocurre para los sistemas de propósito general, al margen de los métodos y técnicas para la formalización de arquitecturas de sistemas distribuidos, existen iniciativas en el dominio sanitario por parte de la industria de implementar software de intermediación que cubra aspectos generales o provea de una auténtica transparencia de distribución. De esta forma un ingeniero de sistemas puede apoyarse en estas arquitecturas (o infraestructuras) para desarrollar su sistema pudiendo heredar la funcionalidad de distribución. Además otro beneficio del uso de software de intermediación es que se asegura la apertura e interoperatividad del sistema de información así como una infraestructura operacional adecuada para la integración de sistemas legados y para el desarrollo de nuevos módulos. Dentro del contexto sanitario encontramos principalmente tres esfuerzos: DHE (*Distributed Healthcare Environment),* OMG HDTF *(Health Domain Task Force)* y HSSP (*Healthcare Services Specification Project).*

6.1. DHE - Distributed Healthcare Environment

DHE [102] es un software de intermediación propiedad de la empresa GESI (Gestione Sistemi per l'Informatica S.R.L.) [103] que surgió en el año 1998 implementando el estándar HISA. Esta arquitectura ha sido ampliamente utilizada en varios proyectos europeos y posee gran versatilidad tecnológica ya que apuesta por una separación entre capacidades de servicio y tecnología subyacente, pudiendo aceptar muchas configuraciones y clientes diferentes. El principal elemento del DHE consiste en un repositorio distribuido para toda la información (clínica, organizacional, de gestión, etc.) de la estructura sanitaria permitiendo que esté disponible dónde y cuándo se necesite para todas las aplicaciones en el sistema de información sanitario. La información almacenada en el DHE puede ser introducida, modificada o recuperada a través de un conjunto de servicios accesibles mediante APIs públicas y estables, independientes de entornos físicos o tecnológicos. Otra de sus características es que realiza especial hincapié en la replicación de transacciones y datos transparentes a las aplicaciones. Por último, a través de interfaces nativas se puede comunicar el DHE con un amplio rango de bases de datos.

6.2. OMG HDTF - Health Domain Task Force

La HDTF [101] es un grupo formado en el año 1996 dentro del OMG heredero de la iniciativa CORBAmed con el objetivo de definir estándares de aplicación en el dominio sanitario siguiendo la orientación a objetos de este organismo. Entre los estándares producidos (y en desarrollo) de este grupo podemos encontrar el servicio de identificación de personas (*Person Identification Service*, PIDS), de consulta de léxico (*Lexicon Query Service*, LECIS), de acceso a observaciones clínicas (*Clinical Observation Access Service*, COAS), de acceso a imágenes clínicas (*Clinical Image Access Service*, CIAS), de acceso a recursos

(*Resource Access Decision*, RAD), de apoyo a la decisión clínica (*Clinical Decision Support Service*, CDSS), de identificación de entidades (*Entity Identification Service*, EIS) y de recuperación, localización y actualización (*Retrieve, Locate, Update Service*, RLUS). Desde el 2005, el HDTF ha reorientado sus esfuerzos a la colaboración con el HL7 en el proyecto HSSP (*Healthcare Services Specification Project*) [104] y sus estándares ya publicados han entrado en el proceso metodológico del proyecto.

6.3. HSSP - Healthcare Services Specification Project

En el año 2005 surge este esfuerzo de colaboración entre organizaciones tan reputadas en el campo de la interoperatividad en el contexto sanitario como son el OMG (a través de su grupo especializado en sanidad HDTF) y el HL7 [105]. Otros colaboradores importantes son el grupo IHE (*Integrating the Healthcare Enterprise*) [106] y la Fundación Eclipse [107]. Su objetivo es la identificación y documentación de especificaciones de servicio, funcionalidades y soporte a la conformidad relevantes para los participantes dentro del ámbito sanitario y con especial esfuerzo en implementaciones en el mundo real. HSSP hereda de esfuerzos anteriores como el HL7DF (*HL7 Development Framework*), OMG MDA, WSDL, XML, UML, SNOMED [108], LOINC [109], arquetipos del OpenEHR, modelo de registro para la continuidad de la asistencia del ASTM [110], etc.

El HSSP se organiza en los siguientes grupos de trabajo:

- Servicios de terminología común 2 (*Common Terminology Services 2*, CTS2): núcleo de servicios para la gestión y mantenimiento de terminologías y vocabularios.

- Servicio de apoyo a la decisión (*Decision Support Service*, DSS): proporciona evaluaciones y recomendaciones automatizadas específicas para cada paciente a partir de sus datos.

- Servicio de directorio de servicios y proveedores sanitarios (*Healthcare Provider and Services Directory Service*, HCPDS): soporta la búsqueda de proveedores y servicios dentro de áreas concretas.

- Servicios de referencia cruzada de identidad (*Identity Cross-Reference Services*, IXS): proporciona un conjunto de capacidades para la gestión y recuperación de información identificativa para varios tipos de entidades (personas, organizaciones, dispositivos, etc.) dentro y entre organizaciones.

- Servicios de privacidad, acceso y seguridad (*Privacy, Access and Security Services*, PASS): conjunto de servicios centrados en la seguridad de las organizaciones sanitarias.

- Servicio de recuperación, localización y actualización y Servicio de identificación de entidad (*Retrieve, Locate and Update Service and Entity Identification Service*, RLUS/EIS). RLUS se encarga de localizar, recuperar y actualizar la información sanitaria dentro y entre organizaciones. Por su parte, el servicio EIS resuelve la identidad de pacientes y entidades a través de los sistemas.

- Ontología de servicios SOA (*SOA Services Ontology*): busca desarrollar una ontología de servicios dentro del paradigma SOA de manera que se puedan obtener los beneficios de la automatización de los procesos.

La visión del HSSP es permitir el uso de servicios (siguiendo el estilo SOA) en el ámbito sanitario mediante el desarrollo de interfaces de servicio estándar y la provisión de guías que expliquen cómo utilizar estos servicios en una SOA. Los servicios que especifica el HSSP son de tres tipos:

1. Servicios de infraestructura: independientes de aplicación proporcionan funciones comunes al sistema como la seguridad, el registro de eventos y la notificación. En el estadio actual del proyecto son este tipo de servicios los que se están especificando.

2. Servicios de negocio: encapsulando capacidades de negocio reutilizables como la gestión de identidad, el registro de pacientes y la documentación clínica.

3. Servicios orientados a actividades específicas sanitarias: a menudo orquestando servicios de negocio. Entre estos servicios podemos encontrar los de telemedicina, los de tratamiento de enfermedades crónicas, los de procesamiento de información y minería de datos clínicos, etc.

Entre los principios de diseño que sigue el HSSP y que están mayoritariamente influidos por su filosofía SOA se pueden encontrar la estandarización de servicios, la abstracción y reusabilidad, la autonomía, el descubrimiento, la composición y el bajo acoplamiento (eliminando dependencias formales entre servicios HSSP). Las interfaces que definen los servicios HSSP se componen de constructores que identifican las semánticas soportadas. En concreto tenemos un significante semántico (*semantic signifier*) el cual identifica el modelo de información computable en la interfaz (nombre del modelo, versión, etc.) y los perfiles semánticos (*semantic profiles*) que restringen los significantes que puede utilizar el servicio. Para la especificación de servicios, el proyecto HSSP utiliza una metodología propia denominada Marco de trabajo para la especificación de servicios HSSP (*HSSP Service Specification Framework,* SSF) que se compone de siete pasos con una duración de entre tres y cinco años. En esta metodología HL7 define en primer lugar los requisitos funcionales para los servicios sanitarios y posteriormente el OMG define las especificaciones técnicas de esos servicios. Esto último se realiza a través de MDA y la especificación de los modelos PIM y PSM de los servicios. Actualmente sólo están completados los servicios RLUS y EIS, mientras que en desarrollo se encuentran los servicios DSS, CTS2 y PASS.

7. Resumen de propuestas y estándares relacionados con formalización de arquitecturas

En la Tabla 2.III se organizan las iniciativas y estándares presentados anteriormente y relacionados con la formalización, diseño e implementación de arquitecturas de sistemas distribuidos tanto a nivel general (ingeniería del software) como en el ámbito de la salud (informática sanitaria). Se añaden algunos esfuerzos extra (caracterizados con un asterisco '*') sólo incluidos como ilustración del esquema

global de este campo de estudio y que en este trabajo de Tesis Doctoral se ha pretendido centrar en la formalización de arquitecturas de sistemas distribuidos en entornos sanitarios.

Propuestas arquitecturales	Propuestas relacionadas	Dominio	
		Ingeniería del software	Informática sanitaria
Marcos arquitecturales	Arquitecturas de empresa (EA)	Zachman DoDAF ISO 1471 FEAF Praxeme „4+1“ TOGAF MEGAF MoDAF MDA RM-ODP SOMF GCM ARIS Archimate	HIS-DF CHF ISO 12967 (HISA)
Modelos arquitecturales	Arquitecturas de referencia Estilos arquitecturales	SOA Agentes Arquitecturas basadas en componentes Arquitecturas conducidas por eventos (event-OA)	OpenEHR* HL7 RIM/CDA* ISO 13606-1* LOINC* SNOMED*
Lenguajes de descripción	Lenguajes de modelado	UML OCL* UML4ODP SysML IDL OWL-S	OpenEHR ADL*
Software de intermediación	Arquitecturas de componentes Arquitecturas de servicios	CORBA Servicios Web Servicios Grid The Globus Toolkit DDS Integración semántica	OMG HDTF HSSP DHE ISO 12967 (HISA)
Procesos de desarrollo de arquitecturas (ver capítulo 3)	Métodos y modelos de diseño y análisis	RUP Métodos Ágiles Harmony TOGAF ADM	HL7 4 SOA HL7 DF*

Tabla 2.III. Resumen de herramientas relacionadas con la formalización de arquitecturas

8. Marco de comparación de métodos y estándares para la formalización de arquitecturas

La heterogeneidad de propuestas, alcances, dominios de aplicación, estado de madurez, etc., de los métodos y estándares para la formalización de arquitecturas es de lo más extensa. Recordando que uno de los objetivos del presente trabajo de Tesis es el diseño de una arquitectura de referencia para sistemas sanitarios distribuidos y abiertos, un paso previo necesario es elegir aquel estándar (o estándares) más adecuados para esta tarea. Para ello, en este apartado se realiza una comparativa entre los diferentes marcos de referencia para la formalización de arquitecturas. Esta comparativa está basada

en dieciocho criterios que pretenden abarcar un conjunto amplio de características. Algunos de estos criterios conducen a valoraciones objetivas (por ejemplo, si existen certificados oficiales de conformidad) mientras que otros son más subjetivos y las puntuaciones quedan a discreción del evaluador. En la figura siguiente pueden verse los criterios incluidos en la comparativa de acuerdo con su carácter objetivo o subjetivo.

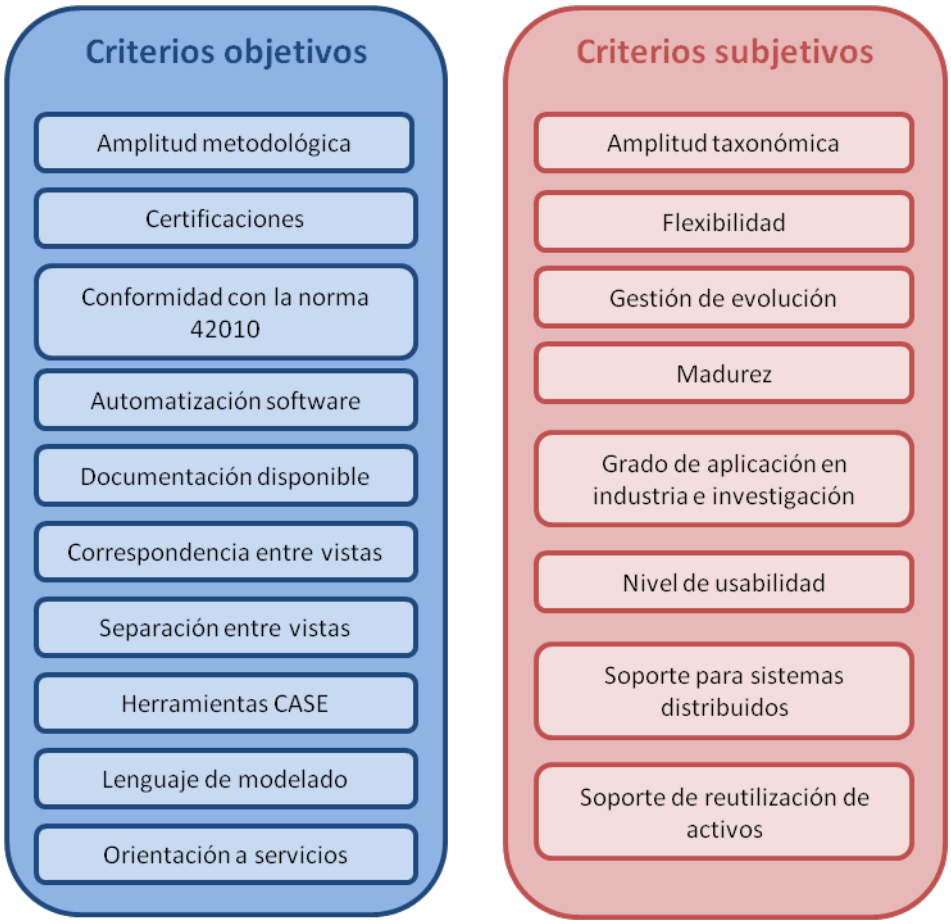

Figura 2.25. Criterios de puntuación objetiva/subjetiva utilizados en la comparativa

A continuación estos criterios serán descritos en detalle y puntuados para cada uno de los marcos de referencia. La puntuación otorgada estará dentro de una escala de 1 a 4, siendo 1 la puntuación más baja (escasa o nula cobertura del criterio evaluado) y 4 la más alta (amplia cobertura del criterio evaluado). Los resultados de la comparativa son susceptibles de servir de ayuda para la elección del marco de referencia más adecuado para cualquier tipo de arquitectura siempre que se establezcan los requisitos específicos para cada caso. Es decir, en función de las necesidades de la arquitectura a formalizar (su objetivo, alcance, dominio…) se otorgará más importancia a unos criterios que a otros ponderando los resultados. En nuestro caso concreto, los resultados de la comparativa permitirán concluir, en base a diversos requisitos que impondremos de acuerdo con este trabajo de Tesis, qué marco (o marcos) de referencia se ajusta mejor a nuestras necesidades particulares.

Las propuestas incluidas en esta comparativa son de propósito general: “4+1”, Archimate, ARIS, DoDAF, FEAF, GCM, MDA, MoDAF, Praxeme, RM-ODP, SOMF, TOGAF y Zachman. Quedan fuera MEGAF y los estándares de exclusiva aplicación en el dominio sanitario. MEGAF está orientado a nivel de meta-arquitectura (es decir, centrado en el desarrollo de marcos arquitecturales y no de las propias arquitecturas como el resto de iniciativas) por lo que no se ha considerado conveniente introducirlo en la comparativa. Los marcos de referencia para arquitecturas sanitarias tienen un alcance más limitado que las propuestas de propósito general y, por tanto, la mayoría de sus características no se ajustan a los criterios generales aquí planteados. Compararlas con el resto de propuestas produciría un sesgo innecesario en los resultados y ninguna sentencia concluyente (por ejemplo, todas las iniciativas sanitarias serían puntuadas con 1 en el séptimo criterio, lo cual no produce resultados con valor pues es una conclusión obvia).

8.1. Aspectos de la comparativa

Criterio 1: Amplitud metodológica

Este primer criterio de comparación valora el proceso de desarrollo que guía la formalización de las arquitecturas de sistemas distribuidos (Figura 2.26). Proporcionar a los arquitectos de sistemas una metodología de trabajo simplifica la aplicación de los marcos de referencia y, en teoría, mejora la eficiencia del proceso de formalización de arquitecturas a través de un conjunto guiado de tareas con ese fin. Se ha considerado que es un criterio objetivo en cuanto es constatable la existencia (o ausencia) de procesos guiados para la formalización de arquitecturas para cada marco de referencia.

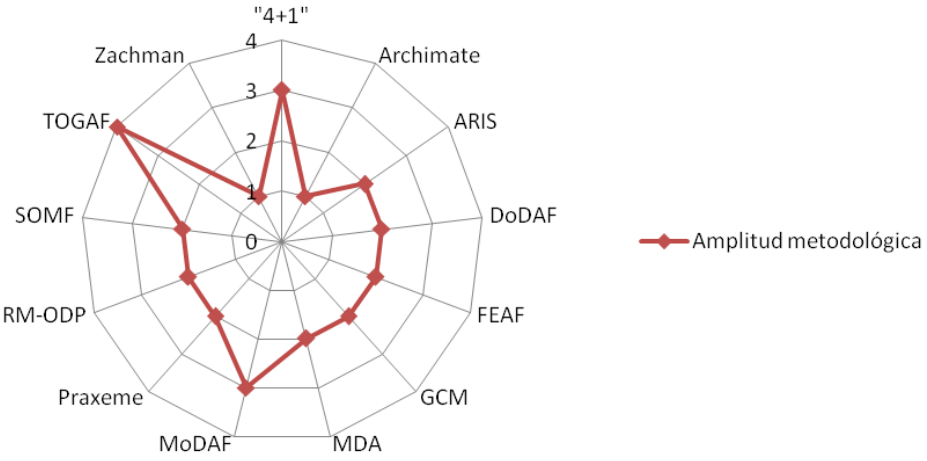

Figura 2.26. Gráfico comparativo de marcos arquitecturales en base a la amplitud metodológica

La puntuación más baja (1) indica que existen pocas (o ninguna) indicaciones sobre los pasos a seguir para formalizar la arquitectura mientras que la más alta (4) establece los esfuerzos que cuentan con un proceso guiado paso a paso exhaustivo; desde la especificación de requisitos hasta la implementación tecnológica. Sólo dos iniciativas para la formalización de arquitecturas de sistemas distribuidos (TOGAF y el modelo de vistas “4+1”) cuentan con metodologías propias. Otros casos (RM-ODP, MDA, etc.) indican

en diferente medida el proceso adecuado de aplicación del marco de referencia dejando al ingeniero de sistemas la decisión última sobre la metodología de desarrollo a adoptar. Sólo dos iniciativas (Archimate y Zachman) incurren en la menor puntuación debido a que no diferencian en absoluto fases (aun desordenadas) del proceso de formalización de arquitecturas.

Criterio 2: Amplitud taxonómica

Este segundo aspecto tiene en cuenta la riqueza del marco de referencia definido en cada una de las iniciativas, esto es, cómo de bien se pueden utilizar para clasificar artefactos arquitecturales (aspectos de negocio, recursos, sistemas de información, actores, etc.). La amplitud taxonómica es una característica de gran relevancia puesto que mide la flexibilidad que poseen los marcos de referencia para acomodar la gran variedad de aspectos y detalles de las arquitecturas de sistemas. Se considera un criterio subjetivo puesto que esa flexibilidad no es cuantificable en términos absolutos sino por comparación entre distintos marcos de referencia.

Se ha resuelto que ninguna de las iniciativas es merecedora de la máxima puntuación pues, en mayor o menor medida, podrían ser extendidas con nuevos aspectos arquitecturales (Figura 2.27). Las taxonomías más completas son las de Zachman, RM-ODP, Praxeme, MODAF y GCM. El resto tienen una puntuación de 2 a excepción del modelo de vistas “4+1” que ha sido el de menor calificación porque, aunque distribuye diversos aspectos arquitecturales en diferentes vistas, cuenta con pocas definiciones formales que permitan una formalización de artefactos arquitecturales rigurosa.

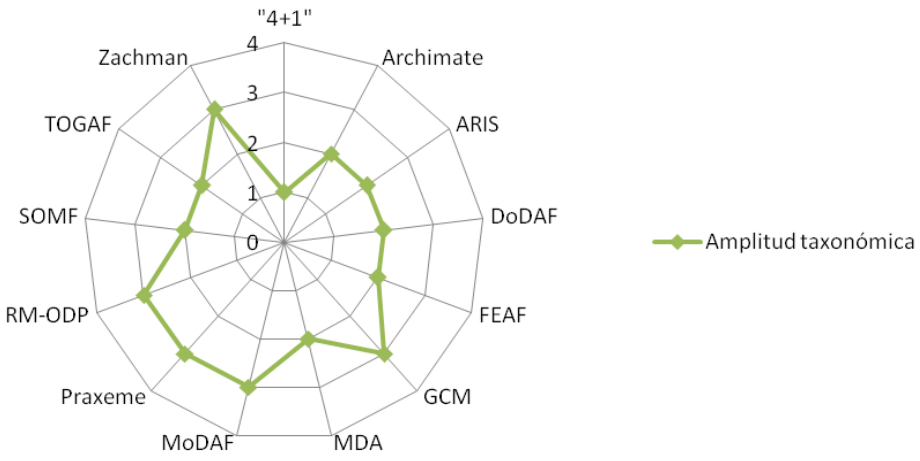

Figura 2.27. Gráfico comparativo de marcos arquitecturales en base a la amplitud taxonómica

Criterio 3: Certificaciones

Algunos de los marcos de referencia cuentan con certificaciones expedidas por organismos competentes para la validación de las arquitecturas que se realicen con ellos. Además en contados casos un individuo concreto puede certificarse como capacitado para desarrollar arquitecturas en base a un método concreto. Este criterio es objetivo y puede resultar interesante considerarlo cuando se desea contar con

conformidad del organismo que promueve el marco de referencia. Este criterio de certificación sólo es aplicable a TOGAF que es la única iniciativa que cuenta con ese tipo de apoyo institucional (Figura 2.28).

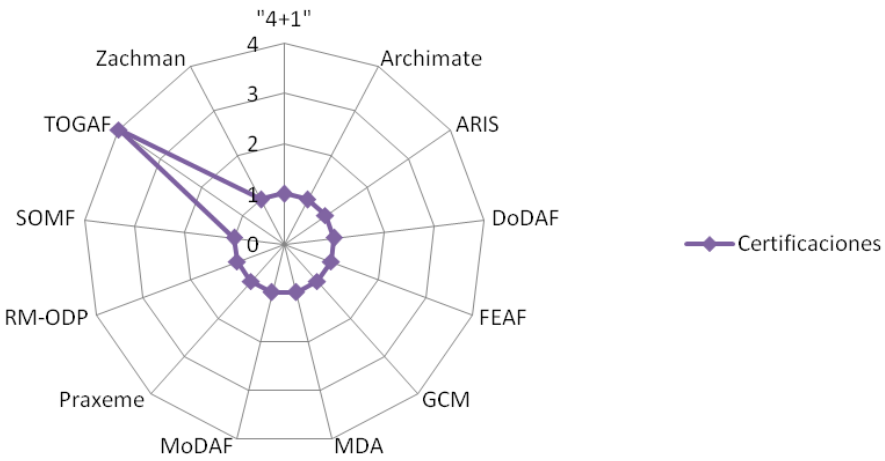

Figura 2.28. Gráfico comparativo de marcos arquitecturales en base a la posibilidad de certificación

Criterio 4: Conformidad con la norma ISO 42010

En el dominio de la formalización de arquitecturas existen varias recomendaciones y guías de buenas prácticas. No todos los marcos de referencia se ajustan a los dictámenes de dichas recomendaciones de manera que un criterio de comparación interesante es la conformidad con las mismas. Debido a que existe un amplio abanico de guías de buenas prácticas se ha optado por elegir una de las más representativas como es la norma ISO 42010 [17]. Cabe destacar que la conformidad con una recomendación como ésta no garantiza la bondad de un método de formalización de arquitecturas pero al menos indica que dicho método se apoya en las mejores prácticas desarrolladas en el sector, lo cual es una consideración a tener en cuenta. La conformidad con esta norma se basa en que la descripción arquitectural, producto de la aplicación de un marco de referencia concreto, debe constar de, al menos, los siguientes elementos:

(a) Documentación de la arquitectura: fecha y estado, organización implicada en el desarrollo, historial de cambios, resumen, alcance, glosario y referencias.

(b) Identificación de participantes e intereses:

   a. Usuarios, clientes, desarrolladores y mantenedores del sistema

   b. Misión del sistema, adecuación con los intereses, factibilidad de construcción, riesgos, capacidad de evolucionar, ser desplegado y mantenido.

(c) Selección de puntos de vista arquitecturales

   a. Nombre, participantes objetivo, intereses a centrar, lenguaje, técnicas o métodos a emplear.

b. Un razonamiento sobre la selección de cada punto de vista

(d) Vistas arquitecturales: identificador, representación del sistema construida con los lenguajes, métodos y técnicas del punto de vista correspondiente, e información de configuración.

(e) Consistencia entre vistas y registro de cualquier inconsistencia existente.

(f) Razonamiento sobre los conceptos arquitecturales utilizados.

En la Figura 2.29 se muestra el grafo comparativo de marcos de formalización de arquitecturas respecto a su conformidad con la norma ISO 42010.

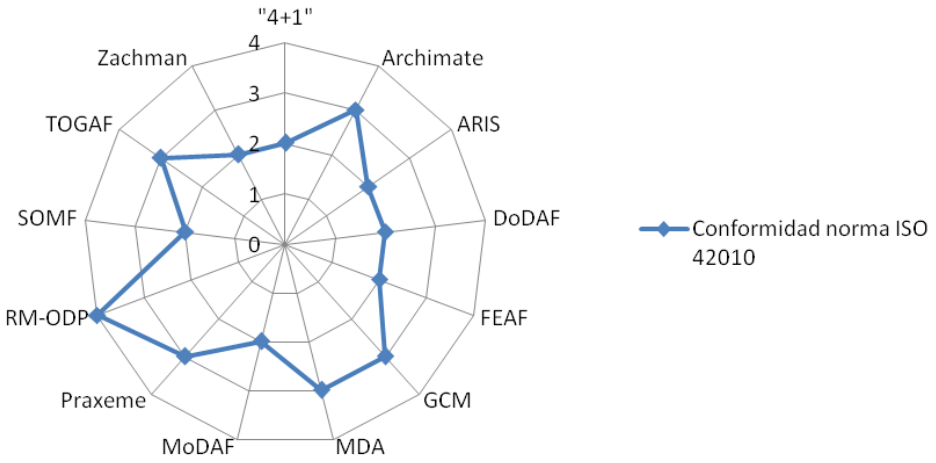

Figura 2.29. Gráfico comparativo respecto de su conformidad con la norma ISO 42010

Muchos son los aspectos que tienen que cubrir las descripciones arquitecturales para ser conformes con esta norma como se deduce del grafo comparativo. El principal punto de disconformidad es la poca profundidad que la mayoría de los marcos de referencia presentan en la descripción de sus vistas. La gran mayoría identifica los propósitos de cada una de las vistas que define pero en general no existen lenguajes específicos para cada una de ellas o existe un único lenguaje de formalización para todas que no considera las diferencias entre ellas. Otro punto frecuentemente obviado en los marcos de referencia es la correspondencia entre vistas (que se abordará como criterio aislado más adelante) y el razonamiento sobre los conceptos arquitecturales utilizados. En este criterio RM-ODP es el método mejor puntuado pues, aunque no es completamente conforme con la norma, sí está por delante del resto de iniciativas.

Criterio 5: Desarrollo automático mediante técnicas software

Este criterio evalúa la posibilidad de aplicación de técnicas software para realizar transformaciones automáticas entre vistas y modelos e incluso, en fases de implementación, entre modelos y código. Es un criterio objetivo en el cual destaca la propuesta de MDA puesto que está basada desde sus fundamentos en las técnicas de ingeniería conducida por modelos soportando la generación automática

o semiautomática de modelos y código. La Figura 2.30 ilustra la comparativa en base a este criterio y se puede ver cómo destaca MDA seguido de otras iniciativas como Praxeme y Archimate que siguen su estela e incluso reutilizan parte de la filosofía del OMG. El resto de marcos de referencia no consideran la automatización de los procesos en su desarrollo aunque en algunos casos se están buscando vías para adoptar estas técnicas (en RM-ODP por ejemplo, a través de su perfil UML [27]) ya que proporcionan bondades reconocidas.

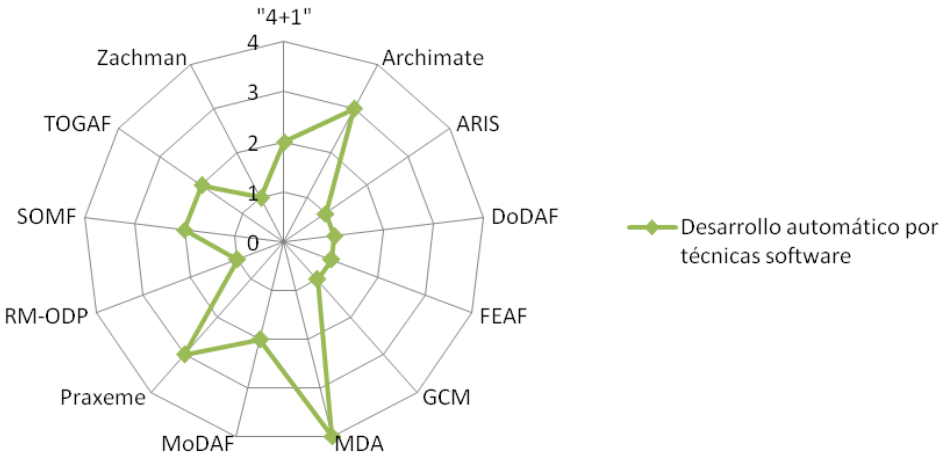

Figura 2.30. Gráfico comparativo de aplicabilidad de técnicas software para desarrollo automático

Criterio 6: Documentación disponible

El criterio objetivo de información o documentación disponible, aunque no directamente relacionado con la formalización de arquitecturas, es esencial en una comparativa puesto que de ello puede depender la decisión de utilizar un marco de referencia u otro. En concreto este criterio se refiere a la cantidad y calidad de información gratuita y de libre acceso que existe sobre cada marco de referencia.

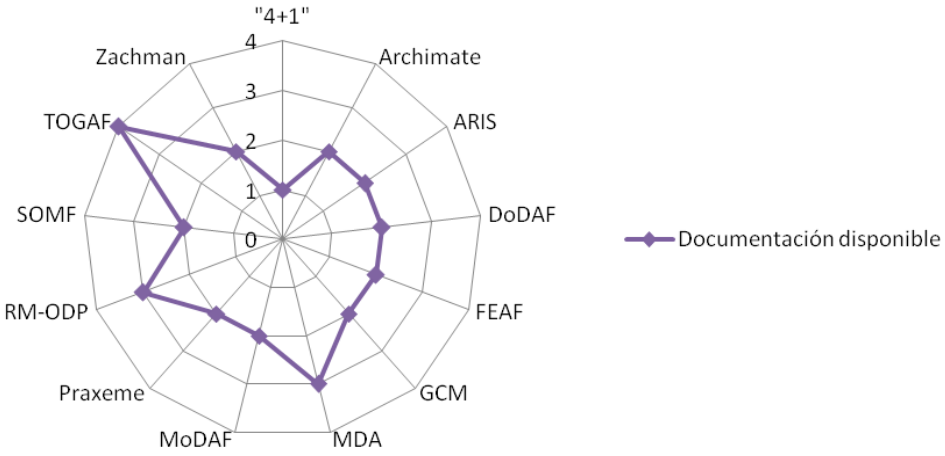

Figura 2.31. Gráfico comparativo respecto a la documentación disponible

El método que proporciona una información más extensa, elaborada y de libre acceso es TOGAF (Figura 2.31). Esto es así siempre que la documentación se utilice para fines académicos o personales (como en

el caso de este trabajo de Tesis). En caso de explotación comercial por parte de una empresa hay que suscribir un acuerdo de licencia comercial anual con *The Open Group*. Le siguen RM-ODP y MDA cuyas documentaciones son completas pero no están tan fácilmente accesibles o tan bien estructuradas. La menor puntuación la recibe el método "4+1" puesto que es muy difícil encontrar una buena definición de sus puntos de vista. Su especificación original sólo proporciona una visión general de su contenido esperado y no se han encontrado definiciones más completas en recursos libremente accesibles.

Criterio 7: Flexibilidad a distintos tipos de organizaciones y dominios de aplicación

En la actualidad los sistemas distribuidos pueden encontrarse en cualquier ámbito de la sociedad y para cualquier tipo de organización por lo que conviene que un método de formalización de arquitecturas sea suficientemente flexible y aplicable a distintos contextos. Este criterio tiene una naturaleza subjetiva puesto que no es sencillo evaluar esta característica de un marco de referencia sin haberlo utilizado en distintos dominios. Desde un punto de vista teórico y a partir de la documentación disponible, la gran mayoría de propuestas han sido valoradas positivamente debido a que ofrecen un marco de referencia general que puede ser adaptado a diversas necesidades (Figura 2.32). Algunas excepciones son MoDAF, FEAF y DoDAF que, si bien son métodos completos y competentes, fueron desarrollados dentro de un ámbito específico de aplicación (organizaciones gubernamentales) lo que hace que su adaptabilidad a otros tipos de requisitos esté limitada.

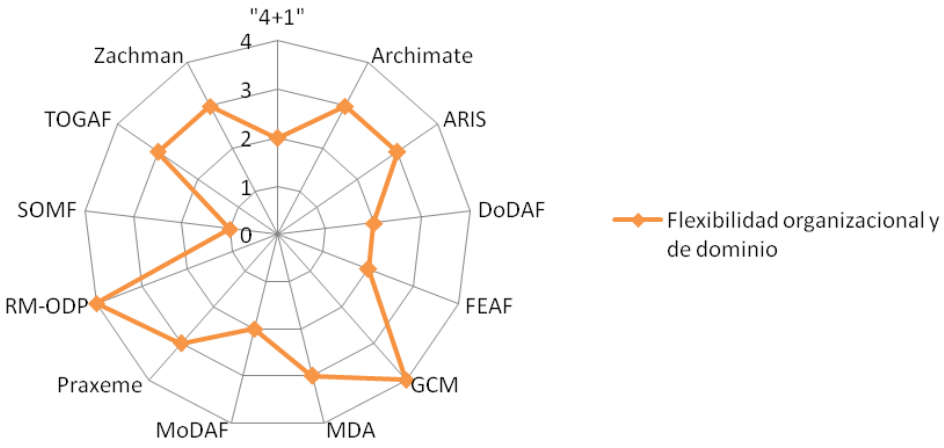

Figura 2.32. Gráfico comparativo respecto a la flexibilidad a distintos tipos de organizaciones

Conviene destacar que un método puede ser muy poco flexible por estar muy centrado en un ámbito específico de aplicación y ser enormemente valioso y eficiente para el mismo. Por ello este criterio sirve como elemento decisorio entre métodos cuando el arquitecto es consciente del dominio de aplicación de su arquitectura, pudiendo cambiar de método en función del alcance de la organización.

Criterio 8: Gestión de evolución de las arquitecturas

Debido a que el proceso de formalización de una arquitectura no acaba con su despliegue sino que debe utilizarse a lo largo de todo el ciclo de vida de un sistema (modificaciones en los requisitos, cambios de

actores, procesos, adaptación a nuevas tecnologías, etc.), es necesario considerar cómo los distintos métodos abordan la evolución de las arquitecturas que especifican. Es un criterio subjetivo en cuanto no se puede cuantificar en términos absolutos el compromiso de cada marco de referencia con la evolución de las arquitecturas.

Siguiendo entonces una valoración relativa por comparación entre marcos, ninguno de ellos consigue una gestión de la evolución completa, principalmente debido a que es un aspecto muy difícil de manejar a priori (Figura 2.33). Sólo destacan de la puntuación media MDA y TOGAF por arriba y Zachman por abajo. TOGAF incluye en su metodología una fase específica para la gestión de cambios de la arquitectura y MDA facilita la adaptabilidad ante cambios gracias a sus lenguajes de modelado formalizados. En el otro extremo está Zachman cuya especificación ignora completamente las posibilidades de evolución de las arquitecturas.

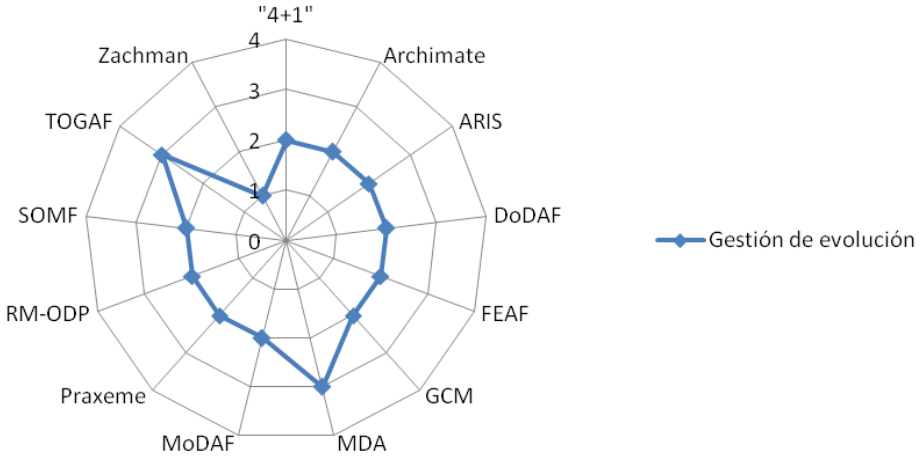

Figura 2.33. Gráfico comparativo respecto a la gestión de la evolución

Criterio 9: Grado de correspondencia entre vistas separadas

Una de las bondades de la formalización de arquitecturas en puntos de vista separados es la simplificación de la tarea, pudiendo relacionar cada punto de vista con un participante implicado en el desarrollo de la arquitectura o con conjuntos de aspectos particulares. Cada punto de vista posee una visión diferente pero todas las vistas que se especifiquen tendrán la misma arquitectura en su foco. Por eso es necesario para establecer una formalización coherente e integrada de una arquitectura el poder identificar las correspondencias entre las distintas vistas, puesto que inevitablemente compartirán elementos comunes.

En la Figura 2.34 se presenta la comparación de los métodos analizados en función al grado de correspondencia entre vistas que permiten especificar. Con menor puntuación tenemos el marco de referencia de Zachman y el modelo de vistas “4+1”. El primero presenta un conjunto extenso de vistas pero las relaciones entre ellas están muy limitadas teniendo incluso vistas cuyos elementos no pueden ser relacionados, y en el caso del segundo hay una carencia total en el sentido de correspondencia entre

las distintas vistas. El método más sofisticado en este aspecto es RM-ODP ya que posee mecanismos formalizados para establecer correspondencias entre sus vistas. Por otro lado, en el caso de TOGAF se asegura la consistencia global entre vistas a través del proceso central de gestión de requisitos de la arquitectura que el resto de fases y vistas deben satisfacer.

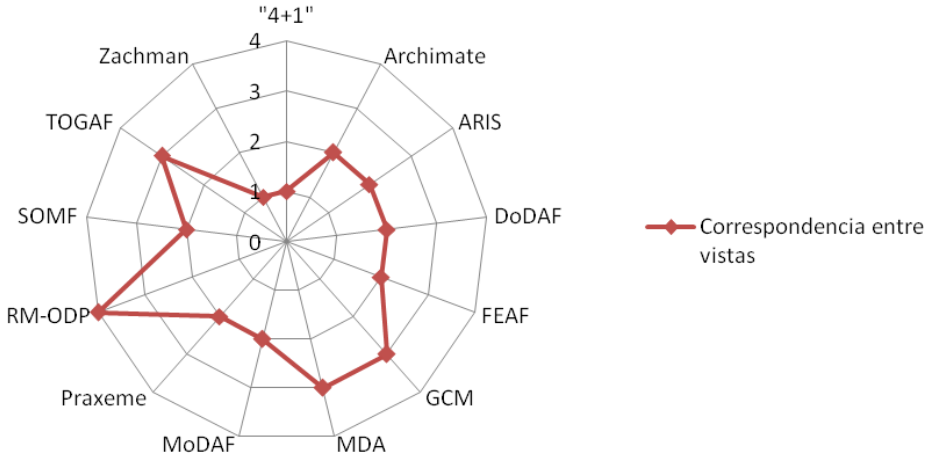

Figura 2.34. Gráfico comparativo en base al grado de especificación de correspondencia entre vistas

Criterio 10: Grado de separación entre vistas

Uno de los aspectos más interesantes de los métodos de formalización de arquitecturas y de la separación en vistas es posibilitar el desacoplo entre los aspectos dependientes e independientes de tecnología. De este modo se facilita la reutilización de especificaciones abstractas para distintas plataformas permitiendo además que los cambios que sufren las arquitecturas con motivo de la evolución de la tecnología de base puedan ser más fácilmente asumidos e incorporados. Los métodos que de forma más eficiente y elaborada separan las vistas relacionadas con la tecnología de las de alto nivel son TOGAF, RM-ODP, Archimate, GCM y MDA (Figura 2.35). Por otro lado cabe destacar que ninguna iniciativa recibe la mínima puntuación. Esto es así porque la práctica de separación entre vistas centrando diferentes aspectos de la arquitectura (entre ellos la tecnología) está muy extendida y aceptada en esta disciplina, y sus bondades derivadas son bien conocidas.

Criterio 11: Herramientas software de apoyo

Este criterio evalúa la cantidad y calidad de herramientas software (exclusivas o genéricas) de apoyo a la formalización de arquitecturas a través de los marcos de referencia analizados. En un proceso tan complejo como es el del desarrollo de arquitecturas la posibilidad de utilizar herramientas que faciliten ciertas tareas es aconsejable por no decir necesario. Un marco de formalización puede estar perfectamente construido, ser exhaustivo y flexible, que si el arquitecto no puede apoyarse en herramientas software para acometerlo nunca alcanzará una gran difusión. Es necesario destacar que en este criterio no se considera que las herramientas software realicen pasos del proceso de desarrollo de

forma automática (como era el caso del criterio 5) sino que se tienen en cuenta todo tipo de herramientas (de modelado, de identificación de requisitos, etc.).

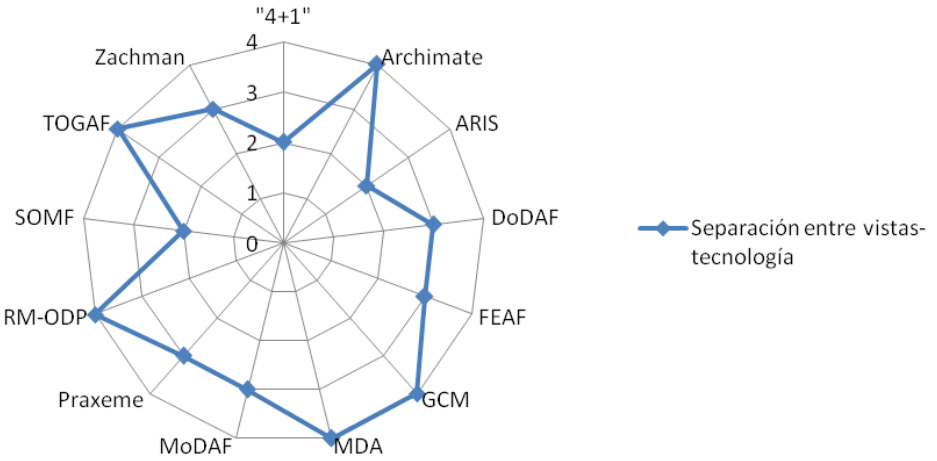

Figura 2.35. Gráfico comparativo en base al grado de separación entre vistas

En la Figura 2.36 se ilustra la comparativa en base a este criterio y podemos ver que MDA (junto a ARIS que se apoya en ella) es la que mejor puntuación obtiene debido a la gran cantidad de esfuerzo que hay detrás de esta iniciativa y que se traduce en gran cantidad de herramientas software que asisten en todo el proceso de desarrollo de arquitecturas. El resto en mayor o menor medida hacen uso de herramientas genéricas que, en algunos casos, presentan complementos o "plugins" específicos para ciertos métodos.

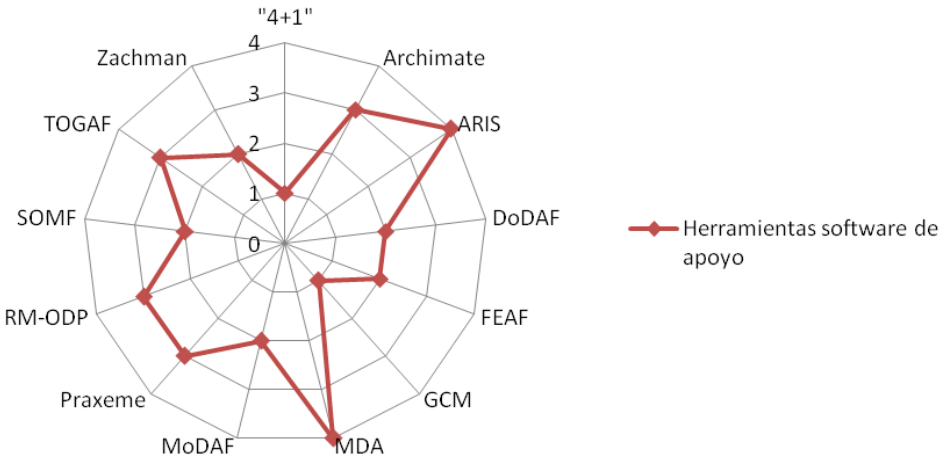

Figura 2.36. Gráfico comparativo respecto al conjunto de herramientas software de apoyo

Criterio 12: Lenguaje de modelado de vistas

Al margen de los conceptos que manejan los distintos marcos de referencia para la formalización de los puntos de vista (y que hemos caracterizado como amplitud taxonómica), cabe preguntarse si se apoyan en algún lenguaje de representación para modelar formalmente esa descripción de la arquitectura en vistas. A través de qué lenguaje de modelado se da forma a las vistas no siempre está especificado

(Zachman o modelo de vistas “4+1”) o, si se indica, existen marcos que adoptan lenguajes de propósito general (RM-ODP, MoDAF o TOGAF) y otros que definen uno propio (Archimate, ARIS o SOMF). Este criterio puntúa mejor a los marcos de referencia que se apoyan en lenguajes de modelado dedicados pues se entiende que serán más efectivos describiendo sus puntos de vista que otros que tienen que hacer concesiones o plantear restricciones sobre lenguajes de modelado de propósito general (Figura 2.37).

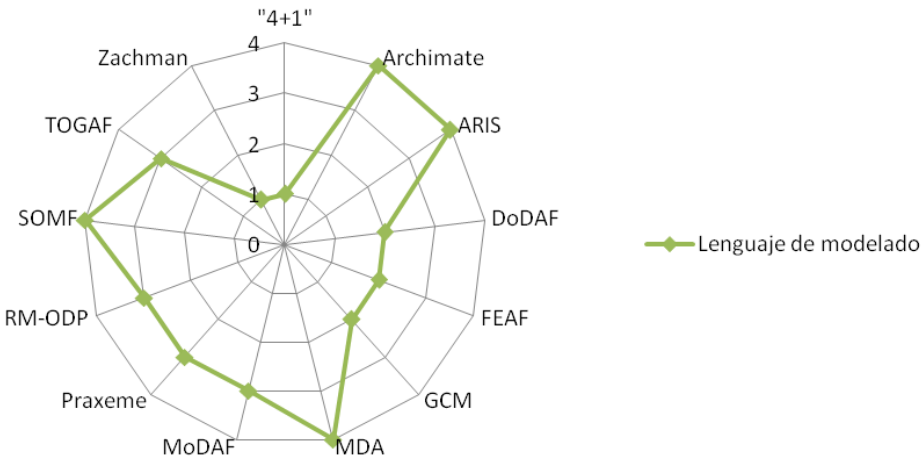

Figura 2.37. Gráfico comparativo en base al lenguaje de modelado de apoyo

Criterio 13: Madurez

El nivel de madurez de un marco de referencia es un criterio subjetivo que indica el recorrido que ha tenido cada propuesta desde su primera publicación. Normalmente esta característica es fruto de procesos de revisión y mejora a través de diferentes versiones. Si bien se asume que todas las iniciativas han necesitado alcanzar cierta madurez en su desarrollo antes de ser por primera vez difundidas, este criterio otorga mayor puntuación a aquellos métodos que han seguido evolucionando y adaptándose a lo largo del tiempo dando lugar a nuevas versiones o modificaciones en sus contenidos. Por tanto, se considera interesante evaluar la madurez de los marcos de referencia porque da una muestra de la estabilidad y robustez de cada propuesta.

En la Figura 2.38 se recogen las puntuaciones sobre este criterio donde destacan Archimate y ARIS por debajo por ser principalmente iniciativas de reciente publicación y el modelo de vistas “4+1” que no ha contado con ninguna mejora desde su publicación. Los marcos de referencia que mayor desarrollo han sufrido son TOGAF (actualmente se cuenta con la novena versión del marco de referencia), RM-ODP (que ha evolucionado con motivo de su adopción por parte de la ISO) y Praxeme y DoDAF que han tenido diferentes versiones desde su primera publicación.

Criterio 14: Nivel de aplicación en industria e investigación

La difusión de un marco de referencia en la industria y la investigación es un importante indicador de las bondades del mismo y su usabilidad. Es necesario tener en cuenta que la aplicación y difusión de un marco es un proceso lento por lo que las iniciativas de reciente publicación no alcanzan puntuaciones altas y esto no se debe a que no hayan sido bien aceptados por los arquitectos de sistemas.

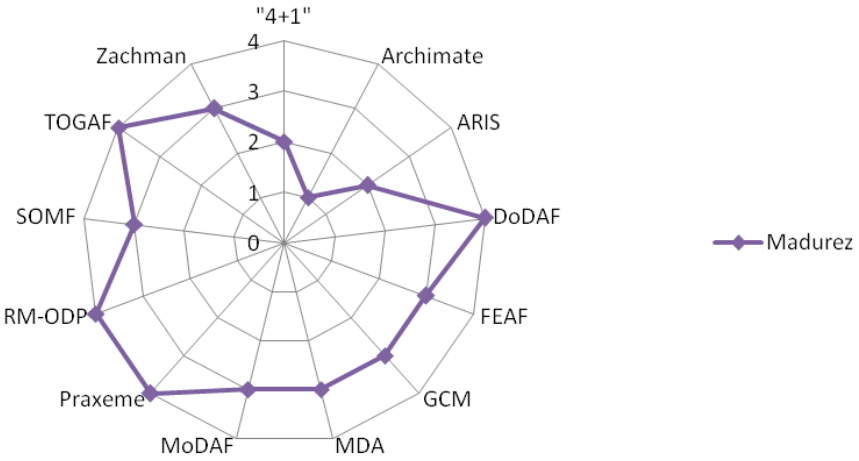

Figura 2.38. Gráfico comparativo en base al nivel de madurez

Como se desprende de la comparativa de la Figura 2.39, MDA es la iniciativa que mayor repercusión y aplicación tiene tanto en el campo industrial como de investigación (prueba de ello es el gran número de herramientas software desarrolladas para su soporte, ver criterio 11) seguido de TOGAF, GCM y RM-ODP. Este último ha alcanzado mayor difusión entre la comunidad científica europea aunque su salto a la industria está haciéndose de forma paulatina. El resto de iniciativas también han sido aplicadas aunque en menor medida no llegando a alcanzar una difusión global sino quedando como métodos de las comunidades de base donde se desarrollaron (por ejemplo MoDAF o DoDAF).

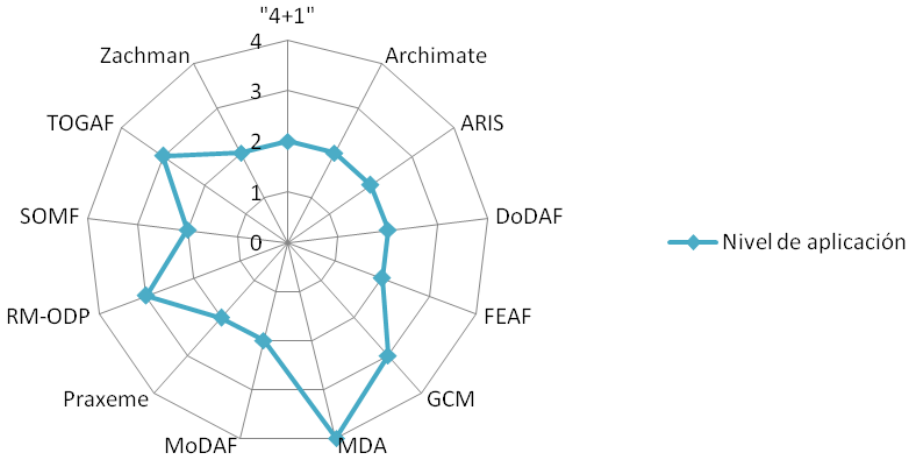

Figura 2.39. Gráfico comparativo respecto al nivel de aplicación en industria e investigación

Criterio 15: Nivel de usabilidad

Este criterio evalúa la facilidad de uso de las diferentes iniciativas. Que un método tenga una puntuación alta indica que puede aplicarse de forma sencilla (quizás porque posee pocos requisitos o un lenguaje poco formalizado o cercano al lenguaje natural) pero no especifica el nivel de bondad de la formalización obtenida. Es un criterio subjetivo pero íntimamente relacionado con varios criterios anteriores (en concreto el 2, 7, 9 y 10). En la Figura 2.40 se muestra el grafo comparativo con respecto a este criterio y se puede comprobar cómo las iniciativas peor valoradas son RM-ODP y GCM pues presentan una compleja separación en vistas de la arquitectura y formalizan las relaciones entre ellas lo que aporta coherencia pero también complejidad en el desarrollo. Las iniciativas más sencillas de aplicar principalmente por lo laxo de su nomenclatura y definición son Praxeme y el modelo de vistas "4+1".

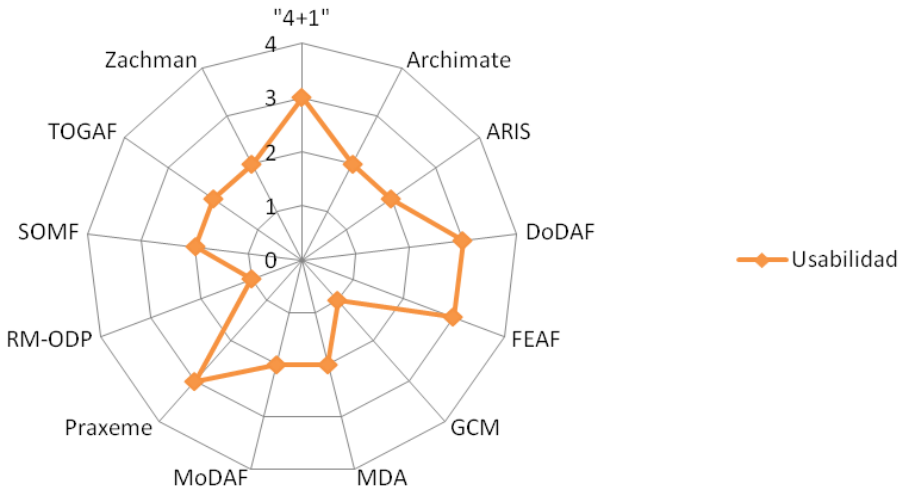

Figura 2.40. Gráfico comparativo de marcos arquitecturales en base al nivel de usabilidad

Criterio 16: Orientación a servicios

Tanto este criterio como el siguiente tienen un carácter más específico y han sido introducidos para evaluar las bondades de los métodos en el campo de las arquitecturas orientadas a servicios (SOA), es decir, el paradigma que se va a utilizar en el presente trabajo de Tesis Doctoral. Por un lado las arquitecturas orientadas a servicios introducen un conjunto de requisitos y una notación especial que es necesario puedan ser incluidos en el método de formalización de arquitecturas elegido. Este criterio es objetivo y puede no ser relevante en la comparativa general pero sí lo es para nuestro trabajo concreto como indicaremos en la ponderación de los resultados al final de este apartado.

El método que mejor representa el paradigma SOA es SOMF que, por su propia definición, está orientado a servicios y ha tomado estado filosofía desde el inicio de su desarrollo (Figura 2.41). Por otro lado y debido a la relevancia de las arquitecturas orientadas a servicios, los métodos más extendidos (MDA, TOGAF y RM-ODP) han analizado las claves para incorporar las nociones SOA a sus arquitecturas. Por ejemplo, en los últimos años RM-ODP ha resuelto un proceso por el que modificaba su segunda parte para alinearse explícitamente con el concepto de servicio [28].

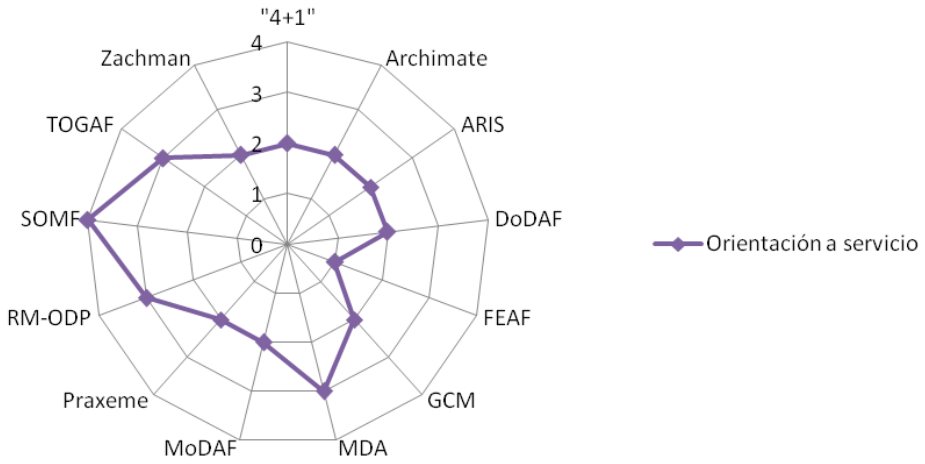

Figura 2.41. Gráfico comparativo de marcos respecto a su adaptación al paradigma SOA

Criterio 17: Soporte para el modelado de características de sistemas distribuidos

Aunque los métodos analizados han sido creados con el objetivo de formalizar arquitecturas de sistemas distribuidos en ocasiones algunas características (como las transparencias de distribución) no son formalizadas de forma adecuada. En la Figura 2.42 se puntúan las distintas iniciativas en este asunto y sólo Praxeme obtiene la máxima puntuación. Tanto esta iniciativa como otras en menor medida (RM-ODP, MDA y SOMF) permiten acoger los requisitos y características de sistemas distribuidos a través de perfiles del lenguaje de modelado UML o utilizando OCL.

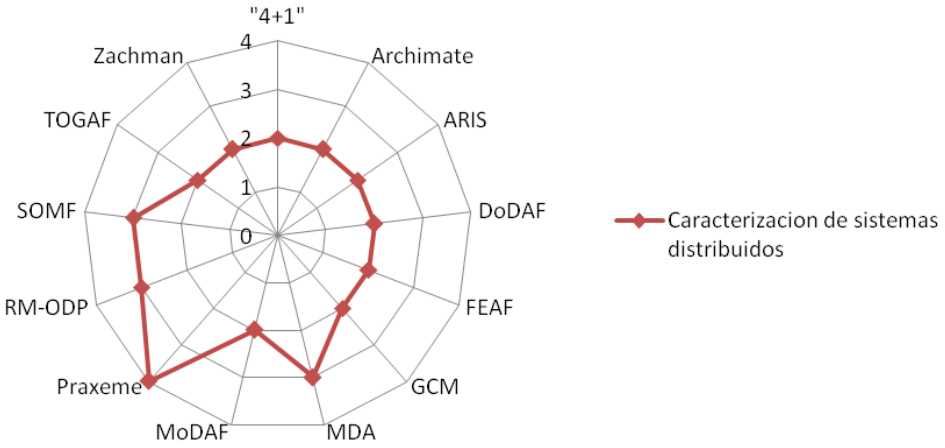

Figura 2.42. Gráfico comparativo en base al modelado de características de sistemas distribuidos

Criterio 18: Soporte para la reutilización de activos

Este último criterio se centra en las capacidades de reutilización de activos arquitecturales (modelos, funciones, etc.) en el desarrollo de las arquitecturas. Esta característica permite facilitar la formalización de arquitecturas pudiendo incluir en el sistema elementos ampliamente desarrollados y testeados que aportan soluciones a problemas recurrentes en las arquitecturas de sistemas distribuidos. Por ejemplo, RM-ODP define funciones básicas que están disponibles para ser incluidas en cualquier tipo de sistema,

y MDA hace lo mismo con modelos de servicios de soporte comunes a todos los sistemas CORBA. La Figura 2.43 ilustra la comparativa en reutilización de activos de los diferentes métodos analizados. Como puede comprobarse, esta característica está muy extendida entre los diferentes métodos y sólo el modelo de vistas "4+1", Zachman y ARIS lo obvian casi por completo.

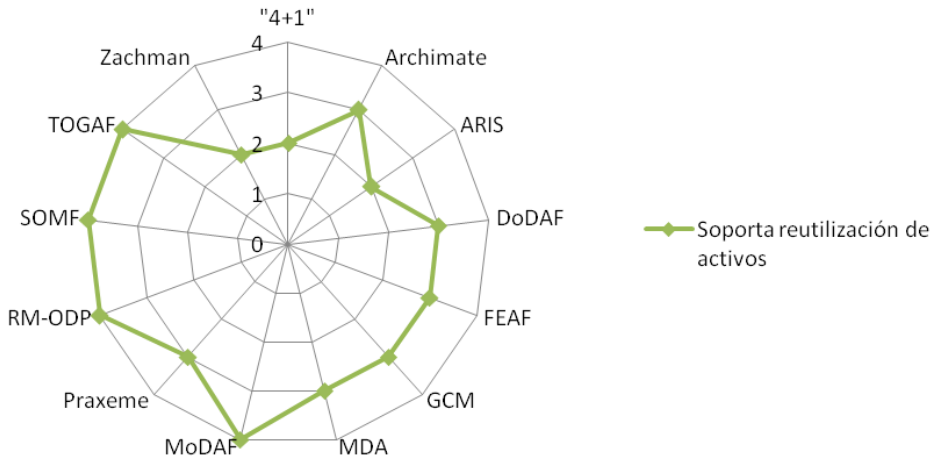

Figura 2.43. Gráfico comparativo en base a la capacidad de reutilización de activos arquitecturales

8.2. Conclusiones de la comparativa y aplicación al escenario de este trabajo

La comparativa realizada es extensa y cubre aspectos muy diferentes de los marcos de referencia. En la Tabla 2.IV se resumen las valoraciones de las distintas iniciativas de acuerdo con los criterios descritos anteriormente. Estos resultados son susceptibles de servir de ayuda para la elección del marco de referencia más adecuado para cualquier tipo de arquitectura siempre que se establezcan los requisitos específicos para cada caso. El presente trabajo de Tesis Doctoral tiene como uno de sus objetivos el contribuir al proceso de formalización de arquitecturas sanitarias distribuidas y abiertas, por ello los criterios más relevantes para nuestro trabajo son:

- Amplitud taxonómica: que mide la extensión de conceptos y detalles del marco de referencia para la formalización de arquitecturas;
- Conformidad con la norma 42010: criterio esencial si queremos seguir la filosofía de la apertura y estandarización;
- Flexibilidad del dominio de aplicación: nuestro dominio de aplicación es el sanitario de tal forma que es necesario que el marco de referencia sea lo más flexible posible a diferentes contextos;
- Grado de correspondencia y separación entre vistas: estos dos criterios se han demostrado esenciales para conseguir especificaciones reutilizables y coherentes; y,
- en menor medida, herramientas software, lenguaje de modelado de vistas, madurez, nivel de aplicación y soporte al modelado de características distribuidas y a la reutilización de activos.

Se ha procedido con un sistema de ponderación de los resultados que otorga pesos desde 1 a 5 a las puntuaciones de las iniciativas. En la Tabla 2.V se muestran los pesos otorgados a cada criterio para nuestro trabajo concreto y en la Tabla 2.VI las puntuaciones finales alcanzadas por los marcos de referencia.

	Criterios																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
"4+1"	3	1	1	2	2	1	2	2	1	2	1	1	2	2	3	2	2	2
Archimate	1	2	1	3	3	2	3	2	2	4	3	4	1	2	2	2	2	3
ARIS	2	2	1	2	1	2	3	2	2	2	4	4	2	2	2	2	2	2
DoDAF	2	2	1	2	1	2	2	2	2	3	2	2	4	2	3	2	2	3
FEAF	2	2	1	2	1	2	2	2	2	3	2	2	3	2	3	1	2	3
GCM	2	3	1	3	1	2	4	2	3	4	1	2	3	3	1	2	2	3
MDA	2	2	1	3	4	3	3	3	3	4	4	4	3	4	2	3	3	3
MoDAF	3	3	1	2	2	2	2	2	2	3	2	3	3	2	2	2	2	4
Praxeme	2	3	1	3	3	2	3	2	2	3	3	3	4	2	3	2	4	3
RM-ODP	2	3	1	4	1	3	4	2	4	4	3	3	4	3	1	3	3	4
SOMF	2	2	1	2	2	2	1	2	2	2	2	4	3	2	2	4	3	4
TOGAF	4	2	4	3	2	4	3	3	3	4	3	3	4	3	2	3	2	4
Zachman	1	3	1	2	1	2	3	1	1	3	2	1	3	2	2	2	2	2

Tabla 2.IV. Puntuaciones de cada iniciativa de acuerdo con los criterios de la comparativa

Criterios	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Pesos	3	5	1	5	2	3	5	3	5	5	4	4	4	4	2	3	4	4

Tabla 2.V. Sistema de ponderación de los resultados de la comparativa para el presente trabajo

RM-ODP	TOGAF	MDA	GCM	Praxeme	SOMF	MODAF
210	205	205	170	183	156	160

Archimate	ARIS	DoDAF	FEAF	Zachman	4+1
162	150	148	141	133	115

Tabla 2.VI. Suma de los resultados ponderados de la comparativa para cada iniciativa

Como se desprende de esta última tabla, RM-ODP es la iniciativa que mayor puntuación suma en los criterios que se han considerado más relevantes para el presente trabajo de Tesis Doctoral. Si bien RM-ODP es la iniciativa que destaca, lo hace por poca diferencia frente a otras como TOGAF o MDA. Por tanto la elección de un marco de referencia para el desarrollo de arquitecturas de propósito general recaería en una de estas tres propuestas, particularizando más concretamente en los requisitos del escenario de aplicación para poder tomar la decisión última.

En el presente trabajo de Tesis Doctoral, la arquitectura a diseñar está orientada en exclusiva para el dominio sanitario y éste es un requisito que debemos tener en cuenta para tomar la decisión. En el Apartado 6 se describieron los marcos arquitecturales más relevantes centrados en el dominio sanitario: HISA, CHF y HIS-DF. La primera se apoya en el marco de referencia RM-ODP, la segunda sigue un desarrollo propietario y la tercera utiliza GCM como base. Combinando esta información con los resultados de la comparativa resolvemos utilizar HISA como referente arquitectural para el diseño de la arquitectura de sistemas distribuidos abiertos para la provisión de servicios del cuidado de la salud y de soporte a la autonomía del ciudadano.

Por último, cabe hacer una mención a la posibilidad de mapear o comunicar arquitecturas especificadas en diferentes marcos de referencia. La existencia de mecanismos que permitiesen esa traducción entre marcos de referencia añadiría flexibilidad a la formalización de arquitecturas puesto que, por un lado, un arquitecto no tendría que limitarse a un marco de referencia único sino que podría compatibilizar varios beneficiándose de sus puntos fuertes y, por otro, facilitaría que especificaciones de arquitecturas realizadas con diferentes marcos de referencia pudieran ser combinadas en arquitecturas más complejas y avanzadas. Lamentablemente, no existe este tipo de mecanismos y la formalización de arquitecturas se debe realizar adoptando un único marco de referencia. Sin embargo, como se presentó en el Apartado 3.7, pueden establecerse correspondencias entre los diferentes marcos de referencia puesto que la mayoría se han desarrollado sobre unos fundamentos comunes (la descomposición del sistema en vistas, la separación de los aspectos independientes de la tecnología de los dependientes, etc.). Resumiendo, aunque no existen mecanismos formales para el mapeo de especificaciones de arquitecturas para diferentes marcos de referencia, un arquitecto puede, con considerable esfuerzo y en base a criterios subjetivos, traducir la especificación de una arquitectura entre distintos marcos de referencia. Obviamente, la existencia de mecanismos de mapeo formales supondría un enorme avance en este campo de conocimiento ya que aportaría flexibilidad a la especificación de arquitecturas e interoperatividad para la combinación de diferentes arquitecturas.

Capítulo 3. Métodos y técnicas de desarrollo

1. Metodologías de desarrollo

En el Capítulo 2 se ha realizado un extenso repaso por los principales paradigmas y tecnologías relacionados con los sistemas distribuidos. Se han estudiado así marcos arquitecturales que proporcionan el cuerpo conceptual necesario para definir unívocamente cada elemento del sistema, lenguajes de descripción que disponen cómo se modela y describe cada sistema, e iniciativas de software de intermediación que facilitan la implementación de estos sistemas proporcionando funciones comunes a todos ellos. Los elementos que se abordan en este capítulo, las metodologías de desarrollo, complementan a los anteriores dando una guía secuencial de pasos a seguir para el desarrollo de cualquier tipo de sistema. En su definición formal, una metodología de desarrollo es una secuencia formal de pasos con el propósito de desarrollar un sistema, cuyos componentes (físicos y lógicos) ofrezcan prestaciones que cubran las necesidades de los usuarios. Entre otras actividades se incluirán la especificación de los requisitos de los usuarios, el diseño de la arquitectura, la elección de tecnologías, etc. En el presente capítulo se estudian varias metodologías de propósito general (RUP, los métodos ágiles, Harmony y TOGAF ADM) así como una exclusiva del dominio sanitario (HL74SOA). Asimismo se presenta un análisis de cómo se relacionan estas metodologías con los principales marcos arquitecturales del capítulo anterior. Además, fruto de estos métodos se ha diseñado una metodología de aplicación de TICs en sanidad la cual se presenta en el correspondiente capítulo de resultados.

1.1. RUP - Rational Unified Process

El método de desarrollo RUP [44] es una de las metodologías más conocidas y aplicadas en el desarrollo de sistemas software y su objetivo es asegurar la producción de software de calidad que satisfaga los requisitos de los usuarios finales ajustándose a una planificación y presupuestos concretos. Describe flujos de trabajo y fases con iteraciones para reducir riesgos en etapas tempranas del ciclo de desarrollo. Cada actividad crea y mantiene modelos UML que se utilizan como representaciones semánticas del sistema software en desarrollo. Cada iteración del proceso produce una versión ejecutable del sistema final que permite una implicación continua del usuario final y su realimentación. Cada ciclo trabaja en una nueva generación del producto y está dividido en cuatro fases consecutivas (Figura 2.22):

- Origen (*inception phase*): en ella se establecen los casos de negocios del sistema y se delimita el alcance del proyecto en esta iteración. Los casos de negocio incluyen criterios de éxito, valoración de riesgos y estimación de los recursos necesarios. Los principales resultados de esta fase son modelos de casos de uso, un plan de proyecto y un prototipo planificado.

- Elaboración (*elaboration phase*): esta fase analiza el dominio del problema, establece los fundamentos arquitecturales, desarrolla el plan de proyecto y se resuelven los principales aspectos que suponen alto riesgo para el proyecto.

- Construcción (*construction phase*): en ella se desarrollan todos los componentes y aspectos de aplicación, se integran en el producto y se prueban todas las características. Esta fase es el proceso

de producción donde lo importante es gestionar los recursos y controlar las operaciones para optimizar los costes, la planificación y calidad. Se lleva a cabo la transición desde el desarrollo de propiedad intelectual de las fases anteriores al desarrollo de productos desplegables en esta fase y la siguiente.

- Transición (*transition phase*): el propósito de esta fase es trasladar el producto software a la comunidad de usuarios. Una vez que el producto es validado por los usuarios finales surgirán nuevos problemas que obligarán a desarrollar nuevas versiones del producto, corregir problemas o cerrar las características que habían sido postergadas a siguientes ciclos de desarrollo.

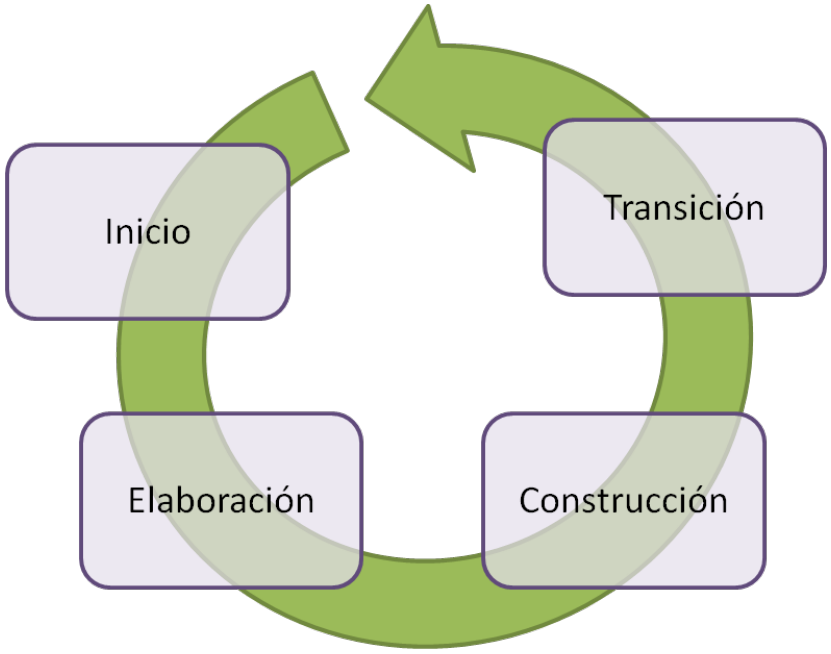

Figura 3.1. Ciclo iterativo y fases del Rational Unified Process

1.2. Métodos ágiles

Otra propuesta de construcción de software que ha tenido gran repercusión ha sido el modelo de desarrollo ágil. Este marco de trabajo promueve también iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. Existen muchos métodos que implementan este modelo y la mayoría minimiza los riesgos desarrollando software en cortos lapsos de tiempo. Una iteración debe durar entre una y cuatro semanas e incluye: planificación, análisis de requisitos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado sino que la meta es tener un prototipo sin errores al final de cada iteración.

Los métodos ágiles determinan además de las fases en las que se descompone una iteración la forma óptima de trabajo para el desarrollo software. La mayoría de los equipos ágiles están localizados en una misma oficina que debe incluir a los revisores, escritores de documentación y ayuda, diseñadores de iteración y directores de proyecto. Debido a que da mayor importancia a las comunicaciones cara a cara que a la documentación generalmente los métodos ágiles son criticados y tratados como

"indisciplinados" por la falta de documentación técnica. Algunos de los métodos ágiles más notables son Scrum [111], *Crystal Clear* [112] o *eXtreme Programming* o XP [113]. Una versión simplificada del proceso RUP y adaptada a los principios de los métodos ágiles es el Proceso Unificado Ágil (*Agile Unified Process,* AUP) [114] que describe de manera sencilla la forma de desarrollar aplicaciones software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

1.3. El proceso de desarrollo Harmony

Aunque los productos software son el objetivo de la mayoría de metodologías de desarrollo existen algunas que cubren un espectro más amplio de productos. Un ejemplo de este tipo es Harmony [115] que cuenta con dos variantes: el proceso Harmony completo (*Full Harmony process*) el cual incluye un detallado proceso de ingeniería de sistemas que precede al desarrollo de cada componente software, y el software Harmony (*Harmony-SW*) centrado exclusivamente en el desarrollo software. En la Figura 2.23 pueden verse las fases que define el proceso Harmony en un ciclo que es un híbrido entre espiral y cascada. Comienza con las fases de ingeniería de sistemas en las que se especifican los requisitos y casos de uso, se define la arquitectura de los componentes, etc. Después de estas fases comienza el ciclo de desarrollo incremental en el cual se irán construyendo prototipos del sistema final añadiendo funcionalidad en cada ciclo y testeándolos por separado. El proceso Harmony utiliza UML como lenguaje de formalización de elementos aunque también permite la utilización de variantes de este lenguaje como SysML o los perfiles UML para DoDAF.

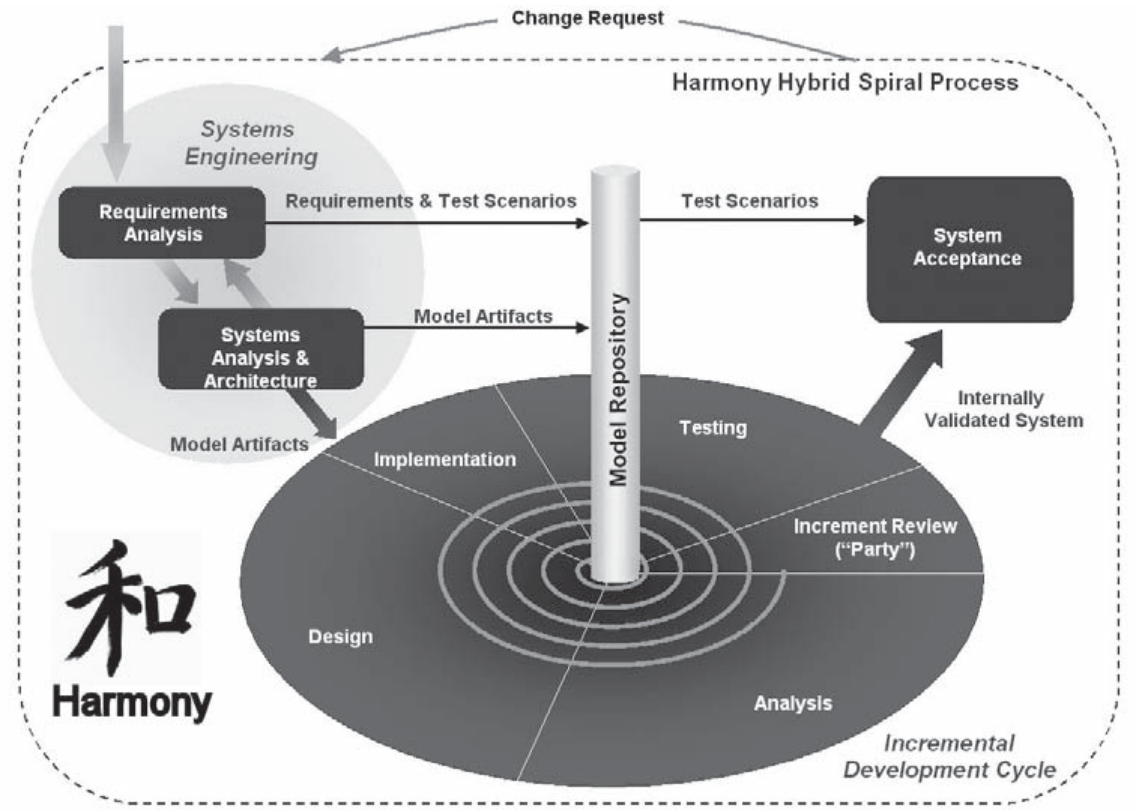

Figura 3.2. Ciclo híbrido cascada-espiral del Harmony Development Process [115]

1.4. TOGAF ADM

Como se describió en el Apartado 3.4.B, el TOGAF ADM es un proceso iterativo que va desde el diseño y desarrollo de las arquitecturas hasta su prueba y realización teniendo como elemento central los requisitos de negocio en todas las etapas de la metodología. Uno de los puntos fuertes de esta iniciativa es que a lo largo del proceso se establecen entregables, artefactos y bloques constructivos que van permitiendo el análisis y la evaluación de forma continua.

1.5. HL74SOA - HL7 for Service Oriented Architecture

Dada la longitud del ciclo de vida del proyecto HSSP, HL7 planteó la iniciativa HL74SOA [116] para crear una metodología de especificación de servicios "provisionales" usando modelos de información HL7v3. Busca definir una propuesta SOA para HL7 y no pretende reemplazar los servicios de HSSP sino construir soluciones específicas que utilicen HL7v3. Este proyecto ha producido una metodología para definir servicios basados en artefactos HL7 así como una propuesta de arquitectura orientada a servicios completa. La armonización entre los mundos SOA y HL7 no se concreta exclusivamente en definir una SOA basada en el modelo de información HL7 sino también en adoptar toda la arquitectura orientada a mensajes de este organismo. La arquitectura que define HL74SOA posee tres niveles de complejidad, cada uno caracterizado por un conjunto de componentes estructurales:

- SOA mínima: registro de servicios, servicio de auditoría y log, almacén persistente de mensajes y proveedores y consumidores de servicios.

- SOA mediada: añade al nivel anterior intermediarios de enrutado y transformación así como un servicio gestor de seguridad y de transacciones.

- SOA dinámica: añade un gestor de políticas, otro de orquestación y uno más de suscripción.

1.6. Armonización entre métodos y estándares

La formalización de arquitecturas consta de marcos de trabajo estándares, metodologías de desarrollo y lenguajes de formalización. Aunque algunos de los estándares cuentan con metodologías propias, en general se deja a discreción del arquitecto elegir el método que considere más adecuado en cada caso. Como ejemplo de conjunción entre las metodologías y los marcos de trabajo descritos en los apartados anteriores, a continuación se establece la correspondencia de modo ilustrativo entre los modelos de referencia RM-ODP y MDA y las metodologías TOGAF ADM y RUP (Figuras 2.24 y 2.25). La correspondencia de las metodologías con el resto de marcos arquitecturales se puede establecer de forma inmediata utilizando las relaciones entre RM-ODP y estos marcos.

Como se puede ver en la Figura 2.24 las fases de la metodología TOGAF ADM en las que se puede hacer uso de los puntos de vista de RM-ODP y MDA son la B, C y D. En la primera, la fase de Arquitectura de Negocio, se identifican los requisitos de negocio clave y se describe el desarrollo de la arquitectura de negocio base en consonancia con la visión de arquitectura de la fase anterior. La arquitectura resultante

de esta fase incluye la estrategia de negocio, directivas, organización e información de los procesos de negocio claves, además de las interacciones entre estos conceptos. Estos elementos finales de la fase pueden ser resueltos a través del punto de vista de empresa del RM-ODP y la vista independiente de computación de MDA. En segundo lugar, la fase C describe el desarrollo de las arquitecturas de sistemas de información identificando y definiendo las consideraciones en el dominio de los sistemas de datos y aplicaciones. La arquitectura de datos define los tipos y fuentes de datos necesarios mientras que la arquitectura de aplicación especifica los tipos de sistemas de aplicación (entendido el término aplicación como un grupo de capacidades lógicas) necesarios para procesar los datos y soportar los procesos de negocio. MDA cubre estos aspectos mediante la vista independiente de plataforma ya que trata de aspectos lógicos del sistema. Por su parte, el RM-ODP podría utilizar el punto de vista de información para resolver la arquitectura de datos TOGAF y el punto de vista computacional para la arquitectura de aplicación. Por último, la fase D de Arquitectura tecnológica está centrada en el mapeo de la arquitectura de aplicación a componentes tecnológicos. Esta correlación de elementos se realiza en RM-ODP a través de los puntos de vista de ingeniería y tecnología mientras que la aproximación MDA utiliza la vista específica de plataforma.

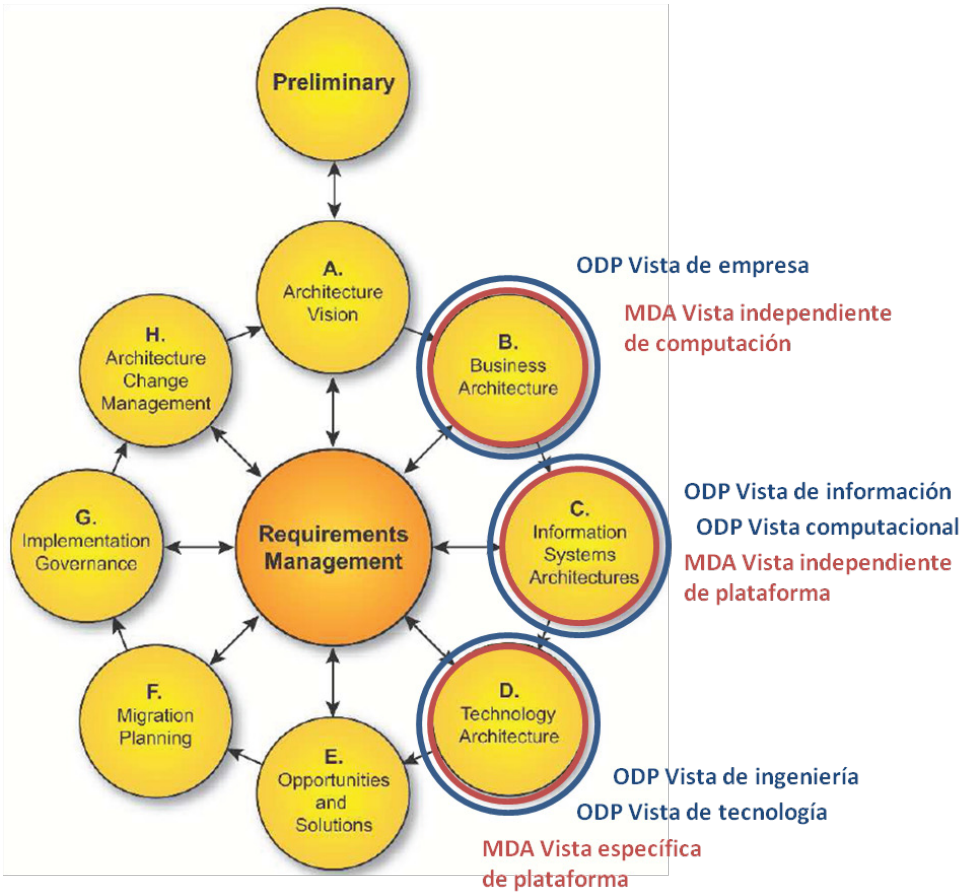

Figura 3.3. Correspondencia entre los marcos RM-ODP y MDA y la metodología TOGAF ADM

La Figura 2.25 indica que las fases del RUP que entroncan con los puntos de vista y modelos de ODP y MDA son las de elaboración y construcción. El propósito de la fase de elaboración es analizar el dominio

del problema y establecer los fundamentos del proyecto que incluyen su alcance y propósito, la funcionalidad general, los requisitos no funcionales y los actores implicados. Esta especificación de requisitos y característica del sistema enlaza en gran medida con el punto de vista de empresa del RM-ODP y la vista independiente de computación de MDA. Por otro lado, en la fase de construcción del RUP todos los componentes y aspectos de aplicación son desarrollados e integrados en el sistema. Por ello en esta etapa el resto de puntos de vista de RM-ODP y las vistas independiente y dependiente de tecnología de MDA son necesarios de cara a especificar completamente el sistema y proceder con la implementación de los componentes físicos y lógicos.

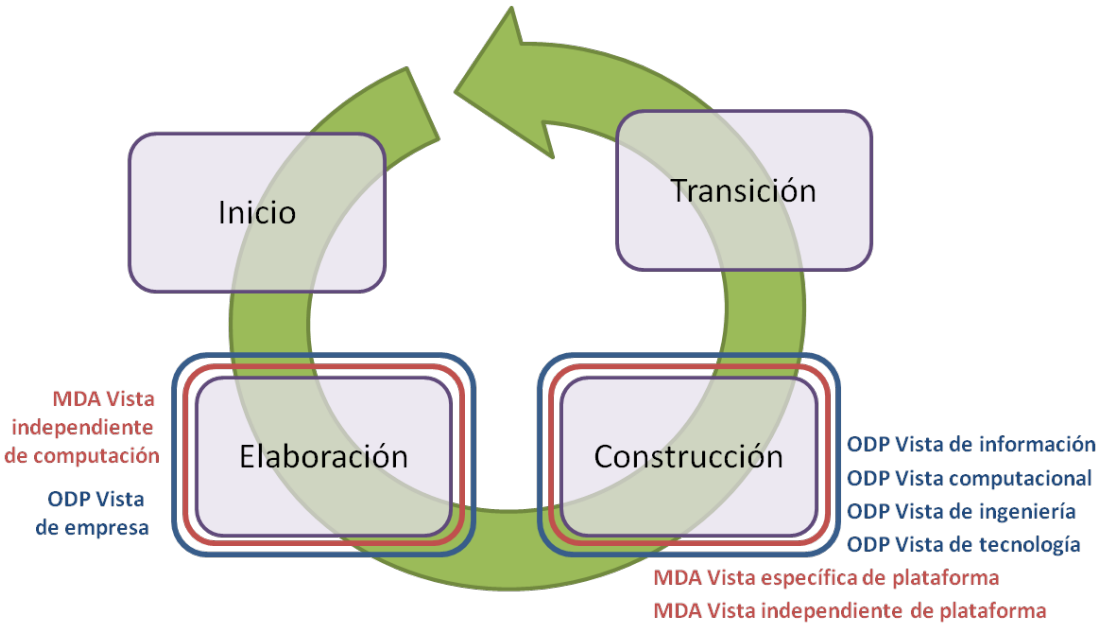

Figura 3.4. Correspondencia entre los marcos RM-ODP y MDA y la metodología RUP

Capítulo 4. El control de acceso y la gestión de privilegios

1. Introducción

El término *seguridad* abarca un conjunto de procedimientos que persiguen minimizar las vulnerabilidades de los recursos y bienes, siendo estos últimos cualquier elemento de valor. Una *vulnerabilidad* es toda debilidad que pudiera explotarse para violar un sistema o la información que éste contiene. En los sistemas distribuidos abiertos los elementos que pueden necesitar protección son la información y los datos, los servicios de comunicación y de procesamiento, y los equipos y sus recursos intrínsecos. Las amenazas a las que se pueden ver sometidos son también numerosas como la destrucción de información y/o de otros recursos, su corrupción o modificación, el robo, supresión o pérdida, la revelación no autorizada o la interrupción de servicios. Desde un principio es necesario identificar las amenazas potenciales de seguridad a las que están sujetos los sistemas de información así como encontrar los mecanismos que hagan posible prevenir, detectar o recuperarse de tales violaciones. Por ello numerosos esfuerzos han sido realizados en el ámbito de la seguridad para, por un lado, definir marcos de referencia normalizados que favorezcan la interoperatividad y colaboración entre sistemas que soporten procedimientos de seguridad y, por otro, implementar y desplegar servicios que provean procedimientos de seguridad en escenarios tecnológicos concretos.

La seguridad cubre un amplio espectro de disciplinas cada una centrada en aspectos concretos de la comunicación o funcionamiento fiables de los sistemas. Así, por ejemplo, para establecer una comunicación segura y confiable son necesarias diversas capacidades que garantizan que una entidad es quien dice ser (autenticación), que nadie puede acceder a la comunicación si no está autorizado (confidencialidad y cifrado), que el mensaje no ha sido manipulado (integridad), etc. Una de las disciplinas más relevantes es la de autorización o control de acceso que se encarga de garantizar que sólo las entidades permitidas pueden acceder a los recursos, limitados además por restricciones en el tiempo y la manera del acceso. El control de acceso ha sufrido una importante evolución debido principalmente a la sofisticación de los sistemas que conlleva un mayor número de requisitos a satisfacer. La distribución de los sistemas es la principal causa del aumento de complejidad de los mecanismos que rigen el control de acceso ya que con la descentralización el entorno se fragmenta en dominios dispares (con administradores, usuarios y recursos en cada uno de ellos), cada uno con sus propias políticas de seguridad y autorización. Es necesario entonces, además de resolver la localización de los elementos necesarios como en cualquier escenario distribuido, satisfacer las restricciones de seguridad impuestas por (y entre) los dominios implicados. Además, la información en la que se basa los agentes decisores para permitir o denegar el acceso a un recurso también ha sufrido una evolución desde la identidad simple del usuario que pretende acceder a la utilización de atributos descriptores para usuarios y recursos, certificados de identidad, características del contexto, etc. Estos procesos evolutivos han dado pie a numerosas propuestas teóricas y prácticas sobre el control de acceso. La evolución de la tecnología plantea, por un lado, nuevos requisitos que exigen mecanismos reguladores del acceso más complejos, y por otro, potencia el desarrollo de nuevas ideas y herramientas que satisfagan dichos requisitos.

El nuevo paradigma de asistencia sanitaria centrada en el sujeto asistido que se propone en este trabajo de Tesis Doctoral descansa sobre una arquitectura de servicios distribuidos y abiertos. Dentro de los numerosos sistemas de soporte a este escenario, los mecanismos de autorización poseen especial relevancia debido a la sensibilidad de los recursos a proteger (relacionados con la salud del ciudadano) y la necesaria administración por parte del sujeto asistido del control de acceso a los mismos. Siguiendo la filosofía de apertura e interoperatividad de la arquitectura, los mecanismos de control de acceso deberán estar fundamentados en estándares y buenas prácticas ampliamente aceptados y utilizados. Debido a que este campo de conocimiento es extenso y heterogéneo, en un primer lugar será necesario hacer un estudio del estado de la técnica para posteriormente establecer un marco de referencia normalizado que nos permita especificar y desarrollar los sistemas abiertos y conformes a estándares. En el presente capítulo se presentan las normas más relevantes en materia de marcos de referencia de seguridad con especial énfasis en las particularizadas en la disciplina de control de acceso a recursos.

2. Marcos normalizados de seguridad

Los marcos de seguridad tratan de la aplicación de servicios de seguridad en un entorno de componentes abiertos, los cuales pueden considerarse cualquier elemento que forma parte de un escenario distribuido desde dispositivos hardware hasta aplicaciones software. Estas normas abordan los elementos de datos y las secuencias de operaciones que se utilizan para obtener servicios de seguridad específicos pero no describen elementos de protocolos ni implementaciones tecnológicas. El fin último de los marcos normalizados de seguridad es sentar las bases para futuras normalizaciones, puesto que proporcionan terminologías y definiciones coherentes de interfaces de servicios abstractos para requisitos de seguridad específicos y clasifican también los mecanismos que se pueden utilizar para satisfacer esos requisitos.

2.1. UIT-T Rec. X.800 – Arquitectura de seguridad para sistemas abiertos

Esta norma (junto con su correspondencia de la ISO) [117] aparece a comienzos de la década de los 90 al amparo del modelo de referencia básico para la interconexión de sistemas abiertos (*Open Systems Interconnection* o modelo OSI) [118]. El objetivo del modelo OSI era permitir la interconexión de sistemas de computación heterogéneos de modo que pudieran lograrse comunicaciones útiles entre procesos de aplicación. En distintos puntos de la comunicación entre sistemas OSI debían establecerse controles de seguridad para proteger la información intercambiada entre los procesos de aplicación. La recomendación X.800 surge con el objetivo de definir los elementos arquitecturales generales relacionados con la seguridad que pueden aplicarse adecuadamente en las circunstancias en que se requiere la protección de la comunicación entre sistemas abiertos. Esta norma establece, dentro del marco del modelo de referencia OSI, directrices y restricciones para mejorar las recomendaciones existentes en el momento de su publicación o para formular nuevas normas con el fin de permitir comunicaciones seguras y proporcionar así un enfoque coherente de la seguridad en OSI. En concreto, la norma da una descripción general de los servicios de seguridad y mecanismos asociados, que pueden

ser proporcionados por el modelo de referencia, y define los puntos, dentro del modelo OSI, en que pueden proporcionarse dichos servicios y mecanismos. El gran valor de esta norma es establecer un marco común para los sistemas distribuidos definiendo los términos implicados y describiendo los servicios relacionados con la seguridad así como los mecanismos que soportan el despliegue de estos servicios.

La recomendación X.800 divide el ámbito de seguridad en cinco servicios principales y establece los mecanismos que pueden realizarlos siempre en un contexto independiente de tecnología.

1. Servicio de autenticación: está encargado de confirmar que una entidad es quien dice ser. Entre los mecanismos que pueden proporcionar este servicio están las técnicas criptográficas, la utilización de información de autenticación como contraseñas y el uso de características y/o propiedades de la entidad.

2. Servicio de control de acceso: proporciona protección contra el uso no autorizado de recursos. Los mecanismos de control de acceso pueden utilizar la identidad autenticada de una entidad o información sobre ella para determinar u aplicar los derechos de acceso de esa entidad. Otros elementos en los que pueden basarse estos mecanismos para la toma de decisiones de acceso son: hora y ruta del intento de acceso, información de los recursos, restricciones de acceso, etc.

3. Servicio de confidencialidad: protege los datos contra la revelación no autorizada y está basado en mecanismos de cifrado.

4. Servicio de integridad: su objetivo es garantizar que los datos no han sido alterados o destruidos de una manera no autorizada. Para garantizar la integridad se pueden utilizar mecanismos como la adición de información suplementaria a los datos que permita verificar la integridad de los mismos en la recepción o la numeración de secuencias.

5. Servicio de no repudio: impide que las entidades implicadas en una comunicación puedan negar haber participado en toda la comunicación o en parte de ella. Este servicio de basa en mecanismos de firma digital, integridad y auditoría.

Servicios y mecanismos	Capa del modelo OSI						
	1-Física	2	3	4	5	6	7- Aplicación
Confidencialidad	X	X	X	X	.	X	X
Cifrado	X	X	.	.	.	.	X
Autenticación	.	.	X	X	.	X	X
Control de acceso	.	.	X	X	.	.	X
Integridad	.	.	X	X	.	X	X
No repudio	.	.	.	.	.	X	X

Tabla 4.I. Relaciones entre servicios y mecanismos de seguridad y las capas del modelo OSI

Debido a que esta norma se enmarca dentro del modelo OSI se describe la correspondencia entre los servicios y mecanismos de seguridad y la capa concreta del modelo en el que se da cada uno. En la Tabla 4.I se muestran algunas de estas relaciones.

2.2. UIT-T Rec. X.810 – Marcos de seguridad para sistemas abiertos

Este marco de seguridad, compartido por las organizaciones ISO e ITU, forma parte de una familia de normas internacionales [119]. Se basa en la norma X.800 anterior e intenta ir un paso más allá en la definición de los servicios y mecanismos de seguridad, extendiendo el marco de seguridad a sistemas abiertos que no sigan el modelo de referencia OSI. Añade nueva terminología y describe los distintos elementos de información que se pueden utilizar en materia de seguridad. Esta norma se divide en las siguientes partes:

- Parte 1 – Visión general (*Overview*): describe los conceptos generales y el resto de disciplinas de seguridad.

- Parte 2 – Autenticación (*Authentication*): describe todos los aspectos de la autenticación aplicables a sistemas abiertos, la relación entre la autenticación y otras funciones de seguridad como el control de acceso y los requisitos de gestión de autenticación.

- Parte 3 – Control de acceso (*Access control*): presenta todos los aspectos de control de acceso de los sistemas abiertos, la relación con otras funciones de seguridad como la autenticación y la auditoría, y los requisitos de gestión de control de acceso. Esta parte de la norma será revisada en profundidad en un apartado posterior.

- Parte 4 – No repudio (*Non-repudiation*): refina y amplía los conceptos de los servicios de seguridad de no repudio descritos en la norma X.800 y proporciona un marco para el desarrollo y la prestación de esos servicios.

- Parte 5 – Confidencialidad (*Confidentiality*): la finalidad de este servicio es proteger la información contra divulgación no autorizada. La confidencialidad es una disciplina implicada en todos los pasos del proceso de gestión de información (extracción, transferencia, almacenamiento…).

- Parte 6 – Integridad (*Integrity*): es la propiedad que indica que los datos no han sido alterados o destruidos de una manera no autorizada. En esta parte de la norma se identifican tipos de mecanismos de integridad y capacidades proporcionadas por cada uno de ellos.

La norma X.810 define el término *dominio de seguridad* el cual comprende un conjunto de elementos bajo una misma *política de seguridad* y está administrado por una sola *autoridad de seguridad* para algunas actividades específicas pertinentes a la seguridad. En las actividades de un dominio de seguridad intervienen uno o más elementos de ese dominio y quizás elementos de otros. Algunos ejemplos de actividades son el acceso a los elementos, el establecimiento y uso de conexiones, etc. La política de

seguridad expresa los requisitos de seguridad de un dominio en términos generales y su aplicación resultará en la identificación de los servicios de seguridad que permitan cumplir las prescripciones impuestas por la política. Generalmente las políticas de seguridad se formulan como principios en lenguaje natural y antes de que estos principios se plasmen en sistemas reales hay que perfeccionar la política de forma que se pueda derivar de ella un conjunto de *reglas de política de seguridad*. Hay dos tipos de reglas para un dominio de seguridad: para actividades dentro del dominio y para actividades entre dominios.

Por otro lado, la norma X.810 especifica qué elementos de información son necesarios en el ámbito general de seguridad. En concreto se establecen cuatro tipos comunes de información de seguridad:

1. Etiquetas de seguridad para indicar la política de seguridad aplicable a un elemento, un canal de comunicación o un ítem de datos. Esta etiqueta también indica, explícita o implícitamente, la autoridad de seguridad encargada de asignar estas etiquetas y un conjunto de atributos de seguridad vinculados al elemento.

2. Valores de comprobación criptográficos para detectar las modificaciones de un ítem de datos. Algunos de estos valores son el sello, la firma digital, la huella dactilar digital, etc.

3. Certificados de seguridad para proteger la información obtenida de una autoridad de seguridad o de una tercera parte confiable, que usarán una o más partes que interactúan.

4. Testigos de seguridad para proteger la información de seguridad que se transmite entre las partes que interactúan.

2.3. ISO/IEC 2382 Tecnología de la información – Vocabulario – Parte 8: Seguridad

La ISO en la década de los 80 publicó la norma ISO/IEC 2382 [120] con el propósito de proporcionar definiciones rigurosas, sencillas y asequibles para todos los agentes implicados en el dominio de las tecnologías de la información. En 1986 se publica la octava parte centrada en la seguridad de los sistemas de información y es en el año 1998 cuando aparece su segunda edición. Esta parte de la norma presenta los conceptos (y sus definiciones) relevantes en el campo de la seguridad e identifica las relaciones entre ellos.

2.4. Marcos de seguridad específicos de organismos

Siendo la seguridad una disciplina transversal y esencial en los sistemas distribuidos, independientemente de la tecnología de implementación o el alcance de tales sistemas, la gran mayoría de organismos de normalización que trabajan en el ámbito de los sistemas distribuidos y la interoperatividad han considerado necesario establecer su propio marco de seguridad. A continuación se describen las iniciativas de cuatro de estas organizaciones: IETF [121], HL7 (*Health Level Seven*) [122], OMG [32] y el grupo IHE [106].

2.4.1. Marco de seguridad de la IETF

En el año 2007 la IETF publicó la segunda versión de su glosario sobre seguridad en Internet, siendo el documento que está actualmente en vigor y cuya referencia es RFC 4949 [123]. Este glosario proporciona definiciones, abreviaturas y explicaciones de la terminología relacionada con la seguridad en los sistemas de información y sirve como modelo conceptual de referencia para todos los procesos de estandarización del IETF.

2.4.2. Marco de seguridad del HL7

La organización HL7 consideró desde sus inicios la necesidad de establecer un marco de seguridad que sirviera de referencia para el desarrollo de estándares [124]. Dada la orientación de este organismo hacia el ámbito de la salud se consideraron sólo aquellos aspectos de seguridad relevantes en los sistemas de información sanitarios así como los servicios y mecanismos relacionados con la implementación práctica de transacciones seguras entre los sistemas. El principal objetivo del grupo de trabajo de HL7 en materia de seguridad es identificar requisitos de usuarios, los servicios necesarios para satisfacer esos requisitos, los mecanismos para proporcionar esos servicios e implementaciones concretas para estos mecanismos. Los servicios de seguridad identificados en el marco de HL7 son:

- Identificación y autenticación: soporta la definición de entidades (como personas o sistemas) y la verificación de que una entidad es quien dice ser.

- Autorización y control de acceso: soporta la concesión de derechos, que incluye la concesión de acceso basado en derechos de acceso y la prevención de usos no autorizados de los recursos.

- Integridad: asegura que los datos o los sistemas software no han sido alterados o destruidos de forma no autorizada.

- Confidencialidad: este servicio asegura que la información no está disponible para individuos, entidades o procesos no autorizados.

- Responsabilidad: asegura que las acciones de una entidad pueden ser inspeccionadas e imputadas a la entidad.

- Disponibilidad: este servicio garantiza que la información está accesible y puede ser utilizada bajo demanda por entidades autorizadas.

- No repudio: soporta la provisión de evidencias con las cuales prevenir que un participante en una acción pueda convincentemente negar su responsabilidad en esa acción.

- Administración: proporciona capacidades para gestionar las políticas de seguridad.

Paralelamente el marco de seguridad identifica los siguientes mecanismos para la provisión de los servicios anteriores: autenticación (por pares y en origen), autorización y control de acceso, integridad, confidencialidad, responsabilidad, no repudio, firma digital, listas de control de acceso, encriptación, claves, registros de auditoría, logs, etc. Para implementar estos mecanismos el marco de HL7 indica varias técnicas o soluciones a nivel tecnológico.

2.4.3. Marco de seguridad del OMG

Otro organismo de estandarización que estableció un marco de referencia en seguridad fue el OMG el cual puso especial atención en la definición de interfaces de objetos que soportaran las actividades relacionadas [125]. El OMG establece que la seguridad se divide en las siguientes disciplinas: confidencialidad, centrada en que la información sea revelada solamente a aquellos usuarios que estén autorizados a acceder a ella; integridad, garantiza que la información es modificada sólo por los usuarios que tienen derecho a hacerlo y sólo de forma autorizada; responsabilidad, los usuarios son responsables de sus acciones relevantes a la seguridad de los sistemas (un caso particular es el de no repudio donde la responsabilidad por una acción realizada no puede ser negada); y disponibilidad, el uso de un sistema no puede ser negado a usuarios autorizados. El ámbito de seguridad es transversal afectando a muchos componentes de un sistema incluyendo algunos que no están directamente relacionados con ella. La seguridad es establecida en un sistema distribuido a través de la provisión de funcionalidades y la imposición de restricciones en la construcción del sistema. Estas funcionalidades son:

- Identificación y autenticación de entidades (usuarios humanos y objetos que necesitan operar bajo sus propios derechos) para verificar que son quien dicen ser.

- Autorización y control de acceso, decidiendo si una entidad puede acceder a un objeto, normalmente utilizando su identidad y/u otros atributos de privilegios de la entidad (tales como rol, grupos, nivel de seguridad) y los atributos de control del objeto accedido.

- Auditoría de seguridad para registrar y monitorizar qué acciones realizan los usuarios y cómo las llevan a cabo.

- Seguridad de la comunicación entre objetos, la cual normalmente se ejecuta sobre niveles de comunicación inseguros. Esto requiere el establecimiento de confianza entre los participantes, lo cual puede requerir de la autenticación mutua de los mismos. También requiere protección de la integridad y, opcionalmente, confidencialidad de los mensajes en tránsito.

- El no repudio proporciona evidencias irrefutables de acciones.

- Funcionalidades para la administración de la información de seguridad (por ejemplo, políticas de seguridad) son también necesarias.

El cuerpo de especificaciones de seguridad del OMG contiene un modelo de referencia que proporciona el marco para la seguridad, una arquitectura que explica cómo ese modelo puedes ser implementado en objetos, un conjunto de interfaces para esos objetos de manera que se puedan manejar las distintas capacidades de seguridad y un conjunto de protocolos centrados en este campo de aplicación.

2.4.4. Privacidad y seguridad en el IHE

La organización IHE surge en 1997 como una iniciativa de profesionales sanitarios y empresas proveedoras cuyo objetivo es mejorar la comunicación entre los sistemas de información que se utilizan en la atención al paciente. Promociona el uso coordinado de estándares como DICOM y HL7 para solucionar necesidades específicas que soporten el cuidado óptimo del paciente y define unos perfiles de integración que utilizan estándares ya existentes para proporcionar un lenguaje común de manera que proveedores y participantes implicados puedan discutir sobre las necesidades de integración en sanidad y las capacidades de integración de los productos del mercado. En un extremo ofrece a los desarrolladores un claro camino de implementación de estándares de comunicación soportados por socios industriales y cuidadosamente documentados, revisados y probados. En el otro extremo, otorga a los compradores herramientas que reducen la complejidad, coste y problemas de gestión de sistemas interoperables.

IHE está organizado en dominios clínicos y/u operacionales como patología anatómica (*anatomic pathology*), cardiología (*cardiology*) y oftalmología (*eye care*) pero de forma transversal a todos estos perfiles se hace necesario ofrecer un modelo de seguridad que proteja los intercambios de información sin disminuir el nivel de interoperatividad [126]. Basándose en la experiencia de los miembros del IHE se han identificado conjuntos comunes de controles de seguridad y privacidad.

1. Controles de responsabilidad: prueban que el sistema protege los recursos de acuerdo a las políticas de seguridad establecidas. Este conjunto de controles incluye auditoría de seguridad, registro de eventos, alertas y alarmas.

2. Controles de identificación y autenticación: permiten comprobar que un sistema o persona es quien dice ser.

3. Controles de acceso: limitan el acceso de una entidad autenticada a la información y funciones para las cuales está autorizada. Estos controles a menudo se implementan usando controles de acceso basados en roles.

4. Controles de confidencialidad: protegen la información sensible de ser expuesta de forma no autorizada.

5. Controles de integridad de datos: prueban que los datos no han sido modificados de forma no autorizada.

6. Controles de no repudio: aseguran que una entidad no puede negar haber participado en una acción.

7. Controles de privacidad del paciente: hacen cumplir las instrucciones relacionadas con información de pacientes específicos.

8. Controles de disponibilidad: aseguran que la información está disponible cuando se necesita.

Entre los perfiles de integración de IHE que pueden ser utilizados para satisfacer requisitos de seguridad y privacidad tenemos: ATNA (*Audit Trail and Node Authentication*), BPPC (*Basic Patient Privacy Consents*), CT (*Consistent Time*), EUA (*Enterprise User Authentication*), XUA (*Cross-Enterprise User Assertion*), DSG (*Document Digital Signature*), XDS (*Cross-Enterprise Document Sharing*), XDR (*Cross-Enterprise Document Reliable Messaging*), XDM (*Cross-Enterprise Document exchange on Media*) y PWP (*Personnel White Pages*). En la Tabla 4.II se muestra la relación entre estos perfiles y los controles de seguridad de IHE donde una 'D' indica una relación directa entre el perfil y la disciplina de seguridad y una 'I' establece una correspondencia indirecta.

	Responsabilidad	Identificación y autenticación	Acceso a datos	Confidencialidad	Integridad de datos	No repudio	Privacidad del paciente	Disponibilidad
ATNA	D	D	D	D	D	D	D	
BPPC				I			D	
CT	D	I				D		
EUA	I	D	I	I		I	I	
XUA	I	D	I	I		I	I	
DSG	D	D			D	D		
XDS				D	D		I	D
XDR				D	D		I	D
XDM			I	D	D		I	D
PWP	I	D	D			I		

Tabla 4.II. Relaciones entre perfiles IHE y controles de seguridad y privacidad

3. Normalización en autorización y control de acceso

Los sistemas distribuidos fomentan la colaboración dinámica de elementos separados y, por tanto, la seguridad es necesariamente un aspecto fundamental integrado en el sistema. Así se darán en

numerosas ocasiones escenarios donde una entidad perteneciente a un dominio de seguridad desee acceder a un recurso alojado en otro dominio. Es por ello que la disciplina de autorización y control de acceso tiene gran importancia en el desarrollo de los sistemas distribuidos y ha centrado numerosos esfuerzos a lo largo de los años. Si además el escenario distribuido se enmarca en un entorno sanitario, los servicios de control de acceso son esenciales ya que se maneja un recurso tan sensible como es la información sanitaria personal de un sujeto asistido.

A pesar de que todas las disciplinas de seguridad descritas en los apartados anteriores merecen ser consideradas al desarrollar una infraestructura de seguridad para sistemas distribuidos, el presente trabajo de Tesis Doctoral se focaliza en la resolución de un marco normalizado en el ámbito de la autorización y el control de acceso que potencie la colaboración e interoperatividad entre sistemas y sitúe al sujeto de la asistencia como administrador del mismo. En este apartado se revisarán las iniciativas de normalización que se utilizarán en el capítulo siguiente como base del marco normalizado de control de acceso. Antes de entrar en la descripción de las distintas normas y recomendaciones es necesario extrapolar algunos de los más importantes principios de diseño para soluciones de control de acceso que han evolucionado en los últimos años [127]:

1. Optimización de las soluciones: el mecanismo utilizado debe ser lo más simple posible. Se considera más adecuado que sea sencillo y robusto en un conjunto de escenarios comunes de control de acceso a que sea complejo y pierda fiabilidad al tratar de cubrir todas y cada una de las eventualidades posibles.

2. Mediación completa: todos los intentos de acceso deben ser explícitamente procesados por los mecanismos de control de acceso siempre. No debe ser posible evitar el control de acceso al acceder a cualquier recurso protegido. Al incluir la disciplina de autorización en la arquitectura normalizada se proporciona “transparencia de acceso” permitiendo que un desarrollador pueda utilizar esos mecanismos sin tener que definirlos.

3. Diseño abierto: todos los algoritmos y mecanismos de seguridad tienen que estar disponibles abiertamente y ser completamente verificables.

4. Mecanismo menos común: los mecanismos que otorgan derechos de acceso no deberían estar compartidos entre diferentes usuarios o aplicaciones. Para cada usuario o aplicación individual un mecanismo diferente o una instanciación del mismo debería ser utilizada.

5. Fallos por defecto: cualquier cosa que no sea explícitamente permitida es denegada por defecto.

6. Separación de privilegios: siempre que sea posible los mecanismos de seguridad deberían verificar condiciones múltiples e independientes antes de otorgar acceso a un recurso protegido.

7. Menor privilegio: cualquier entidad debería obtener el menor conjunto de derechos de acceso posibles para realizar su función o tarea reduciendo así las posibilidades de realizar tareas no autorizadas por un exceso de privilegios.

8. Aceptabilidad del usuario: los mecanismos de seguridad para el acceso a un recurso no deben añadir excesiva carga de trabajo al usuario final o éste buscará caminos alternativos para evitarlo.

9. Resistencia a la confianza: todos los sistemas, actores y entornos deben ser considerados potencialmente inseguros.

10. Aislamiento: todos los mecanismos de gestión de acceso deberían estar aislados de otros sistemas, operando independientemente y estando especialmente asegurados.

Por último es necesario destacar que el control de acceso no puede ser diseñado aisladamente sino que se deben tener en consideración otras disciplinas de seguridad con las que necesariamente interactúa. Así, por ejemplo, los mecanismos de control de acceso que se apoyan en identidades requieren autenticación para asegurar la identidad, las comunicaciones entre los componentes del sistema de control de acceso requieren de protección de la privacidad o la información que se utiliza en el control de acceso, así como las decisiones, pueden ser utilizadas para auditar peticiones de acceso a recursos. A partir de este punto se consideran estas relaciones entre el control de acceso y el resto de disciplinas de seguridad como transparentes, centrándonos en los servicios y mecanismos exclusivos del control de acceso.

3.1. ISO/IEC 10181-3 - Tecnología de la información – Interconexión de sistemas abiertos – Marcos de seguridad para sistemas abiertos: Marco de control de acceso

Esta norma internacional [128] fue desarrollada por la ITU en 1995 (y adoptada por la ISO un año más tarde) con el propósito de definir un marco general para la provisión de control de acceso y ha influido notablemente en los esfuerzos desarrollados posteriormente. El objetivo principal del control de acceso es prevenir del uso no autorizado de un recurso o del uso de un recurso de una manera no autorizada. En concreto este marco de seguridad: define los conceptos básicos para el control de acceso; demuestra cómo se pueden especializar dichos conceptos para permitir algunos servicios y mecanismos de control de acceso habituales; define esos servicios y los correspondientes mecanismos de control de acceso; identifica los requisitos funcionales para los protocolos que soportan los servicios y mecanismos; identifica los requisitos de gestión habituales para soportar los servicios y mecanismos; y se ocupa de la interacción de los servicios y mecanismos de control de acceso con otros servicios y mecanismos de seguridad.

La terminología que define esta norma se nutre de las normas generales de seguridad anteriormente presentadas. Algunos de los conceptos propios del control de acceso que define son:

- **Iniciador** (*initiator*): entidad (por ejemplo, usuario humano o entidad basada en un ordenador) que intenta acceder a otras entidades.

- **Objetivo** (*target*): entidad a la que se puede intentar acceder.

- **Petición de acceso** (*access request*): operación y operandos que forman parte de un intento de acceso.

- **Política de control de acceso** (a*ccess control policy*): conjunto de reglas que definen las condiciones bajo las cuales puede tener lugar un acceso.

- **Información contextual**: información relativa a o derivada del contexto en el que se realiza una petición de acceso (por ejemplo, la hora del día, ubicación del iniciador o el trayecto de comunicación).

- **Información de control de acceso** (*access control information*, ACI): cualquier información utilizada con fines de control de acceso, puede incluir la información contextual o darse ésta por separado.

- **Información de decisión de control de acceso** (*access control decision information*, ADI): la parte de la ACI (probablemente toda) que está disponible para que la función de decisión de control de acceso tome una determinada decisión.

- **Función de decisión de control de acceso** (*access control decision function*, ADF): función especializada que toma las decisiones de control de acceso aplicando reglas de política de control de acceso ante una petición de acceso y la ADI (de iniciadores, objetivos, peticiones de acceso o que se deriva de una decisión anterior).

- **Función de imposición de control de acceso** (*access control enforcement function,* AEF): función especializada que forma parte del trayecto de acceso entre un iniciador y un objetivo en cada petición de acceso y que impone la decisión tomada por la ADF.

En la Figura 4.1 puede verse una ilustración sobre las entidades y funciones básicas involucradas en el control de acceso mientras que en la Figura 4.2 se muestra la información manejada por la ADF.

La AEF asegura que el iniciador sólo realiza accesos permitidos sobre el objetivo tal como determina la ADF. Cuando un iniciador realiza una petición para realizar un acceso al objetivo, la AEF informa a la ADF que se requiere una decisión para tomar una determinación. A fin de realizar esta decisión se facilita a la ADF la petición de acceso (como parte de la petición de decisión) así como: la ADI del iniciador (ADI que se deriva de la ACI vinculada con el iniciador), la ADI del objetivo (ADI que se deriva de la ACI vinculada con el objetivo), y la ADI de la petición de acceso (ADI que se deriva de la ACI vinculada con la petición de acceso). El resto de la información que se aporta a la ADF está constituida por las reglas de política de

control de acceso (a partir de la autoridad del dominio de seguridad de la ADF) así como cualquier información contextual necesaria para interpretar la ADI o la política.

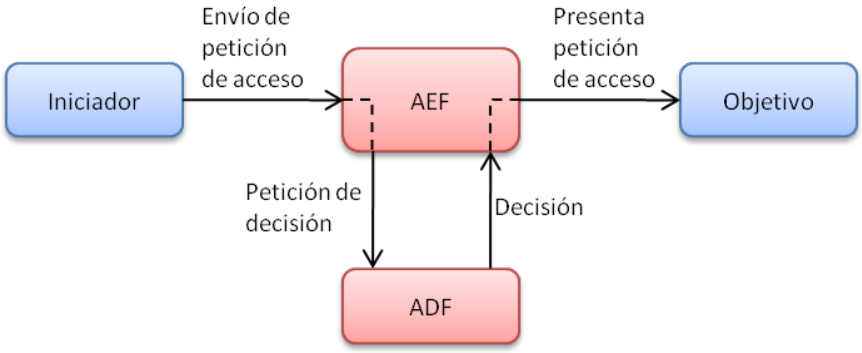

Figura 4.1. Funciones de control de acceso fundamentales según la norma ISO 10181-3 [128]

En base a la información aportada (y posiblemente a partir de ADI de decisiones previas) la ADF concluye en permitir o denegar al iniciador el intento de acceso al objetivo. La decisión se traslada a la AEF que permite que la petición de acceso pase al objetivo o bien tome otras medidas oportunas. Frecuentemente las sucesivas peticiones de acceso realizadas por un iniciador sobre un objetivo están relacionadas como cuando una aplicación realiza una conexión a otra aplicación y solicita varios accesos utilizando la misma ADI. En algunas peticiones de acceso sucesivas enviadas en la conexión puede ser necesario proporcionar a la ADF ADI adicional para que se permita el acceso solicitado. Otras veces la política de seguridad puede exigir que determinadas peticiones de acceso entre uno o más iniciadores y uno o más objetivos estén sujetas a restricciones. En estos casos, la ADF puede utilizar ADI retenida procedente de decisiones anteriores que incluían múltiples iniciadores y objetivos para tomar la decisión sobre una petición de acceso en concreto.

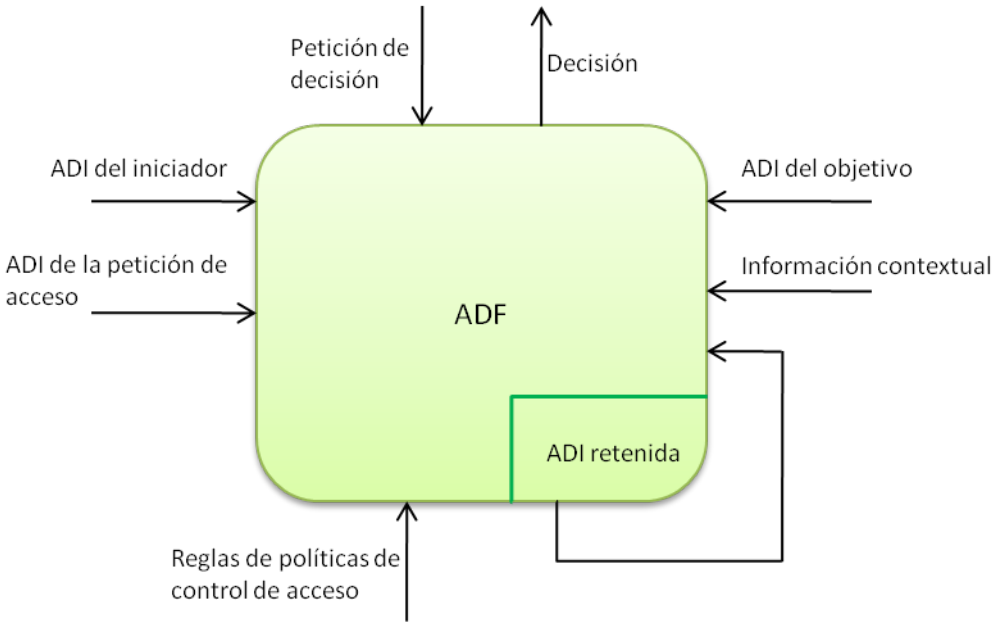

Figura 4.2. Ilustración de la ADF según la norma ISO 10181-3 [128]

Al margen de la función de control de acceso propiamente dicha, esta norma identifica otro conjunto de actividades relacionadas con la autorización como son el establecimiento de representaciones de políticas de control de acceso (de lenguaje natural a reglas procesables por los sistemas) y de ACIs (estructuras de datos y sintaxis), la actividad de otorgar y enlazar ACI a los iniciadores y objetivos (esto lo realiza una autoridad del dominio de seguridad), hacer disponible ADI para la ADF, y modificación, revocación y envío de ACIs. Como estándar abierto que es, esta norma no especifica cómo se realizan o implementan estas funciones y actividades, sólo las identifica.

Un punto a destacar de este esfuerzo normativo es la consideración de componentes distribuidos para realizar las funciones de control de acceso. Así una AEF o una ADF pueden estar compuestas de uno o más componentes de control de acceso y las funciones de control de acceso pueden estar distribuidas entre dichos componentes. Las funciones de control de acceso básicas presentadas anteriormente son independientes de consideraciones relativas a la ubicación de los componentes, a las comunicaciones entre ellos o a su posible distribución. Otra contribución de esta norma es la clasificación de los mecanismos de control de acceso, los cuales se componen de un esquema de control de acceso (por ejemplo, basado en listas de control de acceso, capacidades, etiquetas y contexto) y de los mecanismos de apoyo que proporcionan la ADI a la ADF para dicho esquema. Los esquemas típicos de control de acceso pueden definirse en términos de la ACI vinculada con el iniciador o vinculada con el objetivo de la forma siguiente:

a) **Esquema de capacidades**: cada iniciador tiene vinculado una ACI compuesta de parejas objetivo-tipo de operación, indicando explícitamente qué operaciones puede realizar sobre qué objetivos. La ACI vinculada con el objetivo es simplemente la identidad del mismo.

b) **Esquema basado en la etiqueta**: se consideran “acreditaciones” y “clasificaciones” como la ACI vinculada con el iniciador y la ACI vinculada con el objetivo respectivamente. La decisión de acceso parte de un mapeo entre acreditaciones y clasificaciones.

c) **Esquema de lista de control de acceso**: aquí la ACI vinculada con el iniciador es su identidad y cada objetivo tiene una ACI compuesta por las parejas identidad de iniciador-tipo de operación, indicando qué operaciones puede realizar cada iniciador sobre él.

d) Normalmente, junto con otros esquemas de control de acceso, se utilizan reglas relativas a información contextual aunque éstas deban utilizarse solas para crear un esquema de control de acceso. La información contextual puede formar parte de la ACI vinculada con el iniciador, la ACI vinculada con la petición de acceso o la ACI vinculada con el objetivo, o bien puede ponerse a disposición de la ADF con independencia de otra ACI.

3.2. Control de acceso basado en roles (RBAC) y en atributos (ABAC)

Uno de los mecanismos de control de acceso más populares durante muchos años ha sido el basado en roles (*Role-Based Access Control*, RBAC). La investigación en seguridad de bases de datos introdujo por primera vez la noción de roles para controlar el acceso como una forma de agrupar permisos y facilitar la administración de seguridad. En 1992 Ferraiolo y Kuhn [129] acuñaron el término RBAC y esfuerzos posteriores desarrollaron los primeros modelos que ponían de manifiesto sus características y explicaban su funcionamiento. Estos trabajos fueron la fuente de la que surgieron los esfuerzos NIST (*National Institute of Standards and Technology*) RBAC [130] y ANSI INCITS (*American National Standards Institute – International Committee for Information Technology Standards)* 359-2004 [131]. En este paradigma de control de acceso, las decisiones están basadas en roles que poseen los usuarios individuales por ser parte de una organización. Los derechos de acceso a los recursos se agrupan por roles y el uso de un recurso particular está restringido a los usuarios autorizados que posean un rol específico. En RBAC a los usuarios se les otorgan roles en función de sus competencias o responsabilidades en la organización y esos roles especifican las operaciones que un usuario puede realizar. Debido a que la asignación de un rol a un usuario puede ser fácilmente revocada o modificada, la administración de la seguridad y la gestión de privilegios se simplifica considerablemente; los roles pueden ser actualizados sin necesidad de actualizar los privilegios de cada usuario.

Uno de los principios que deben cubrir los mecanismos de control de acceso es el de menor privilegio (*least privilege*) que estipula no otorgar a un usuario más privilegios de los estrictamente necesarios para realizar su trabajo minimizando así los riesgos de accesos no autorizados. Este requisito es muy costoso y difícil de alcanzar en RBAC principalmente porque esta solución tiende a dar uniformidad en los permisos que se asignan a los usuarios (reutilizando roles) y el principio de menor privilegio plantea estudiar el trabajo particular de cada usuario y otorgarle los permisos adecuados. La satisfacción de este requisito depende por tanto de la implementación concreta RBAC y la complejidad del escenario en el que se trabaje.

El modelo de referencia actual de RBAC se compone de cuatro elementos:

1. Núcleo RBAC: define la colección mínima de elementos y sus relaciones (Figura 4.3) e incluye las funciones de asignación entre usuarios y roles así como entre roles y permisos. Un rol es un mecanismo para nombrar relaciones muchos-a-muchos entre usuarios y permisos. Cada sesión es un mapeo entre un usuario y un subconjunto activo de roles que son asignados al usuario. Este componente siempre debe aparecer pues es la esencia del paradigma mientras que el resto son opcionales y dependen de la implementación.

2. RBAC jerárquico: añade relaciones para soportar jerarquías de roles. Es una forma de estructurar los roles para reflejar las líneas de autoridad y responsabilidad de una organización. La jerarquía de roles define una herencia de permisos entre roles de manera que si r_1 hereda de r_2, entonces todos los privilegios de r_2 son también privilegios de r_1.

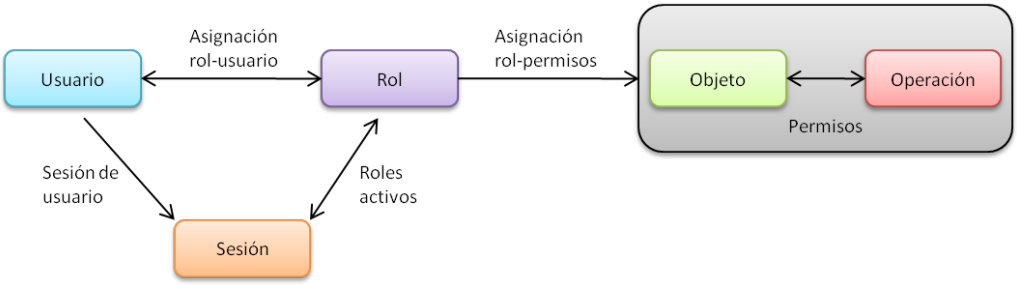

Figura 4.3. Elementos y relaciones del paradigma RBAC [131]

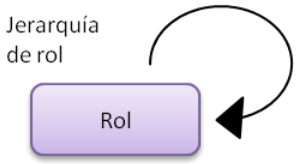

Figura 4.4. Relaciones entre roles en RBAC jerárquico [131]

3. Separación estática de deberes (*Static Separation of Duties*, SSD): añade relaciones de exclusividad entre roles con respecto a la asignación a usuarios, hablando entonces de mecanismo RBAC restringido (*Constrained RBAC*). Éste permite hacer cumplir las políticas de las organizaciones que resuelven los conflictos de interés evitando que los usuarios se excedan en su autoridad. La separación estática define roles disjuntos, es decir, roles que no pueden ser asignados a un usuario simultáneamente. Este mecanismo trabaja sobre todo el espacio de permisos del usuario en tiempo de asignación de un nuevo rol.

4. Separación dinámica de deberes (*Dynamic Separation of Duties*, DSD): a diferencia del anterior este mecanismo trabaja en el espacio de permisos del usuario para cada sesión, es decir, los roles activados al inicio de la sesión. Permite mayor flexibilidad puesto que un usuario puede tener dos roles conflictivos y usarlos en diferentes sesiones lo que no estaría permitido por la separación estática. Además este componente de RBAC favorece enormemente el principio del menor privilegio ya que para cada sesión se pueden activar dinámicamente los roles estrictamente necesarios.

La especificación funcional de estos cuatro componentes del modelo RBAC se ilustra en la Tabla 4.III. Las funciones identificadas se agrupan en las relacionadas con la administración (creación y mantenimiento de elementos y relaciones RBAC), las de soporte (creación y mantenimiento atributos RBAC en sesiones de usuarios), las de revisión (realizan consultas sobre los elementos), y las de revisión avanzada (consultas durante las sesiones).

Funciones	Núcleo RBAC	RBAC jerárquico	Static SD	Dynamic SD
Adminis-tración	*AddUser* *DeleteUser* *AddRole* *DeleteRole* *AssignUser* *DeassignUser* *GrantPermission* *RevokePermission*	*AddInheritance* *DeleteInheritance* *AddAscendant* *AddDescendant*	*AssignUser* *CreateSsdSet* *AddSsdRoleMember* *DeleteSsdRoleMember* *DeleteSsdSet* *SetSsdSetCardinality*	*CreateDsdSet* *AddDsdRoleMember* *DeleteDsdRoleMember* *DeleteDsdSet* *SetDsdSetCardinality*
Soporte	*CreateSession* *DeleteSession* *CheckAccess* *AddActiveRole* *DropActiveRole*	*CreateSession* *AddActiveRole*	----------	*CreateSession* *AddActiveRole*
Revisión	*AssignedUsers* *AssignedRoles*	*AuthorizedUsers* *AuthorizedRoles*	*SsdRoleSets* *SsdRoleSetRoles* *SsdRoleSetCardinality*	*DsdRoleSets* *DsdRoleSetsRoles* *DsdRoleSetCardinality*
Revisión avanzada	*RolePermissions* *UserPermissions* *SessionRoles* *SessionPermissions* *RoleOperationsOnObject* *UserOperationsOnObject*	*RolePermissions* *UserPermissions* *RoleOperationsOnObject* *UserOperationsOnObject*	----------	----------

Tabla 4.III. Descripción funcional de los componentes de RBAC

Aunque el estándar ofrece un marco de referencia para el desarrollo de sistemas RBAC también presenta algunas limitaciones que han sido criticadas en la literatura [132] y contra las que se proponen las siguientes mejoras:

- La eliminación de sesiones del núcleo de RBAC que pasarían a formar parte de un componente separado y opcional debido a que existen sistemas RBAC que no utilizan sesiones. La noción de sesión en el núcleo fundamental del estándar restringen el modelo RBAC innecesariamente.

- Posibilitar las sesiones con un único rol activo y no sólo aquellas con múltiples roles. En ocasiones se puede elegir sesiones con un único rol activo para satisfacer el principio de menor privilegio sin tener que recurrir a separaciones dinámicas de deberes. Se simplificaría así el desarrollo y la lógica de comprobación de conflictos entre roles.

- El proceso de especificar la estructura de roles inicial de una gran organización es una tarea costosa que requiere enormes cantidades de tiempo.

- RBAC es demasiado inflexible para organizaciones dinámicas (diferentes de las organizaciones clásicas) o dominios altamente evolutivos.

Además de las anteriores críticas, se concluyó que un sistema RBAC puro proporciona un soporte inadecuado para atributos dinámicos como la información de contexto (por ejemplo, la hora del día) que

pueden ser utilizados para tomar decisiones. Para soportar atributos dinámicos un sistema RBAC debe sufrir una "explosión de roles" generando gran cantidad de roles separados por los distintos valores que pueden tomar los atributos de manera que se cubrieran todas las configuraciones de atributos posibles.

Recientemente ha aumentado el interés por un modelo de control de acceso que en lugar de basar las decisiones en los roles de los usuarios lo hace en atributos. Este paradigma, denominado ABAC (*attribute based access control*) [133], sugiere que los atributos y reglas podrían reemplazar al modelo RBAC o, al menos, hacerlo más simple y flexible. Debido a que los atributos que utiliza ABAC pueden referirse a cualquier tipo de información (usuarios, recursos, contexto, etc.), este modelo surge como una propuesta de solución que cubre las limitaciones de RBAC y agrupa además los distintos mecanismos utilizados hasta la fecha. ABAC aún no cuenta con un modelo estándar propiamente dicho aunque la norma XACML que se estudiará en el apartado siguiente ha sido tomada como referente de este nuevo paradigma.

ABAC está siendo utilizado progresivamente en solitario pero existen esfuerzos por combinarlo con RBAC intentando obtener lo mejor de ambos paradigmas. Estas iniciativas pueden resumirse en tres nuevos modelos de control de acceso:

1. Roles dinámicos: se utilizan atributos para elegir el rol del sujeto. Se cuenta con una estructura de roles tradicional pero se permite cambiar de rol dinámicamente a través de atributos como la hora del día o la ubicación del usuario.

2. Centradas en atributos: entre la gama de atributos que se utilizan existe uno llamado *rol* con un valor concreto en lugar de un conjunto de permisos. La desventaja es que existe la noción de rol pero no las implicaciones y bondades de relacionarlo directamente con privilegios del usuario.

3. Centradas en roles: se utilizan atributos para las reglas de restricciones de permisos (RBAC restringida) de manera que se pueden reducir los permisos que otorgan los roles al sujeto pero nunca se pueden ampliar. Se pierde el dinamismo que aporta ABAC.

3.3. ITU-T X.1142 - Lenguaje extensible de marcas para el control de acceso (XACML)

El lenguaje extensible de marcas para el control de acceso (XACML) [134] es un tipo de lenguaje XML que se utiliza para expresar políticas de control de acceso. Aunque su núcleo es el lenguaje la norma va más allá de la sintaxis del mismo y define un modelo de control de acceso ABAC, el contexto del modelo, las reglas de procesamiento y perfiles de aplicación del lenguaje a escenarios concretos. XACML es un lenguaje de propósito general, flexible y con potencial para especificar y controlar el cumplimiento de preferencias de control de acceso. Actualmente existen numerosas implementaciones propietarias y de código abierto [135] [136]. Los principales términos que emplea esta norma para describir los elementos involucrados en el control de acceso son:

1. **Atributo**: característica de un sujeto, recurso, acción o entorno a la que puede hacerse referencia en un predicado u objetivo.

2. **Autoridad de atributo** (*attribute authority*, AA): entidad que vincula atributos a entidades.

3. **Punto de administración de políticas** (*Policy Administration Point*, PAP): entidad lógica donde se mantienen y gestionan las políticas de control de acceso.

4. **Punto de cumplimiento de políticas** (*Policy Enforcement Point*, PEP): entidad lógica que lleva a cabo el cumplimiento de las políticas y decisiones de control de acceso.

5. **Punto de decisión de políticas** (*Policy Decision Point*, PDP): entidad lógica que toma las decisiones de control de acceso en respuesta a peticiones que le pasan los PEP de usuarios intentando acceder a recursos protegidos.

6. **Punto de información de políticas** (*Policy Information Point*, PIP): entidad lógica que actúa como fuente de valores de atributos.

7. **Decisión de autorización**: resultado de evaluar la/s política/s aplicable/s notificado por el PDP al PEP. Es una función cuyo resultado es "*Permit*" (permitir), "*Deny*" (denegar), "*Indeterminate*" (indeterminado) o "*NotApplicable*" (no aplicable) además de un conjunto opcional de obligaciones.

8. **Función de servicio de contexto** (*Context handler*): entidad lógica que convierte el formato propio de las peticiones de decisión en la forma canónica de XACML y convierte de las decisiones de autorización al formato propio de la respuesta.

9. **Obligación**: operación definida en una política o conjunto de políticas que debería realizar el PEP cuando ejecuta una decisión de autorización (por ejemplo, auditoría).

La política completa aplicable a una determinada petición de decisión puede estar compuesta por varias reglas o políticas individuales. XACML define los tres elementos de política más generales: <*Rule*>, <*Policy*> y <*PolicySet*>. El elemento <*Rule*> contiene una expresión booleana que puede evaluarse pero que un PDP no puede tratar aisladamente. Por lo tanto este elemento no tiene por objeto formar la base de una decisión de autorización por sí mismo. Sólo podrá existir aisladamente en un PAP de XACML donde puede constituir la unidad básica de gestión y ser reutilizada en múltiples políticas. El elemento <*Policy*> contiene un conjunto de elementos <*Rule*> y un procedimiento especificado para combinar los resultados de su evaluación. Es la unidad de política básica utilizada por el PDP y, por tanto, se considera que es la base de una decisión de autorización. El elemento <*PolicySet*> contiene un conjunto de elementos <*Policy*> u otros <*PolicySet*> y un procedimiento especificado para combinar los resultados de su evaluación, así es el medio normalizado para combinar políticas separadas en una sola política combinada. XACML define varios algoritmos de combinación que definen los procedimientos para tomar

una decisión de autorización en función de los distintos resultados de la evaluación de un conjunto de políticas. Algunos ejemplos de algoritmos de combinación de políticas son “denegación-prioritaria” (si se encuentra una sola regla o política que da como resultado “Denegación” entonces se deniega el acceso completo) o “permiso-prioritario” (si se encuentra un solo “Permiso” se permite todo el conjunto de políticas aun cuando hayan dado como resultado “Denegación”).

En cuanto a los atributos sobre los que basar la decisión de autorización, aunque lo normal es utilizar características del sujeto distintas de su identidad (como el rol en el caso RBAC) el lenguaje XACML permite utilizar cualquier tipo de atributo y no sólo del sujeto sino también atributos del contexto o del recurso. Como aspecto complementario si el recurso puede representarse como documento XML, XACML proporciona facilidades para utilizar el contenido del mismo como atributos de base para tomar la decisión de acceso. El modelo de flujo de datos, es decir cómo se relacionan los elementos, se ilustra en la Figura 4.5. En algún momento previo al acceso, los PAP escriben políticas y conjuntos de políticas que se ponen a disposición del PDP. Estas políticas o conjuntos de políticas representan la política global para un determinado objetivo. En un determinado instante el solicitante de acceso envía una petición al PEP y éste la transmite a la función de servicio del contexto en el formato original posiblemente con atributos de los sujetos, el recurso, la acción y el entorno. La función de servicio del contexto a su vez construye una petición XACML y la envía al PDP el cual solicita, si fuese necesario, a la función de servicio del contexto otros atributos del sujeto, el recurso, la acción y el entorno. Estos atributos adicionales los solicita la función de servicio del contexto a un PIP. Éste los recupera posiblemente desde distintas fuentes y los devuelve a la función de servicio. Ésta envía los atributos solicitados al PDP que evalúa la política y devuelve el contexto respuesta (con la decisión de autorización) a la función de servicio del contexto. Esta respuesta es traducida al formato original del PEP y devuelta al mismo. Finalmente el PEP satisface las obligaciones (si las hubiera) y, si se autoriza el acceso, el PEP permite acceder al recurso. Si no se obtiene autorización, deniega el acceso.

El lenguaje XACML se puede utilizar en distintos entornos ya que la base del lenguaje está aislada del entorno de aplicación por el contexto XACML que se define mediante un esquema XML que constituye la representación canónica de entradas y salidas del PDP. Los atributos señalados mediante un ejemplar de política XACML pueden presentarse como expresiones XPath sobre el contexto o como denominadores que identifican el atributo especificando el sujeto, el recurso, la acción o el entorno y su identificador, el tipo de datos y (es facultativo) el expedidor. La implementación debe convertir la representación del atributo en el entorno de aplicación y su representación en el contexto XACML, aunque esta norma no especifica cómo ha de hacerse esta conversión.

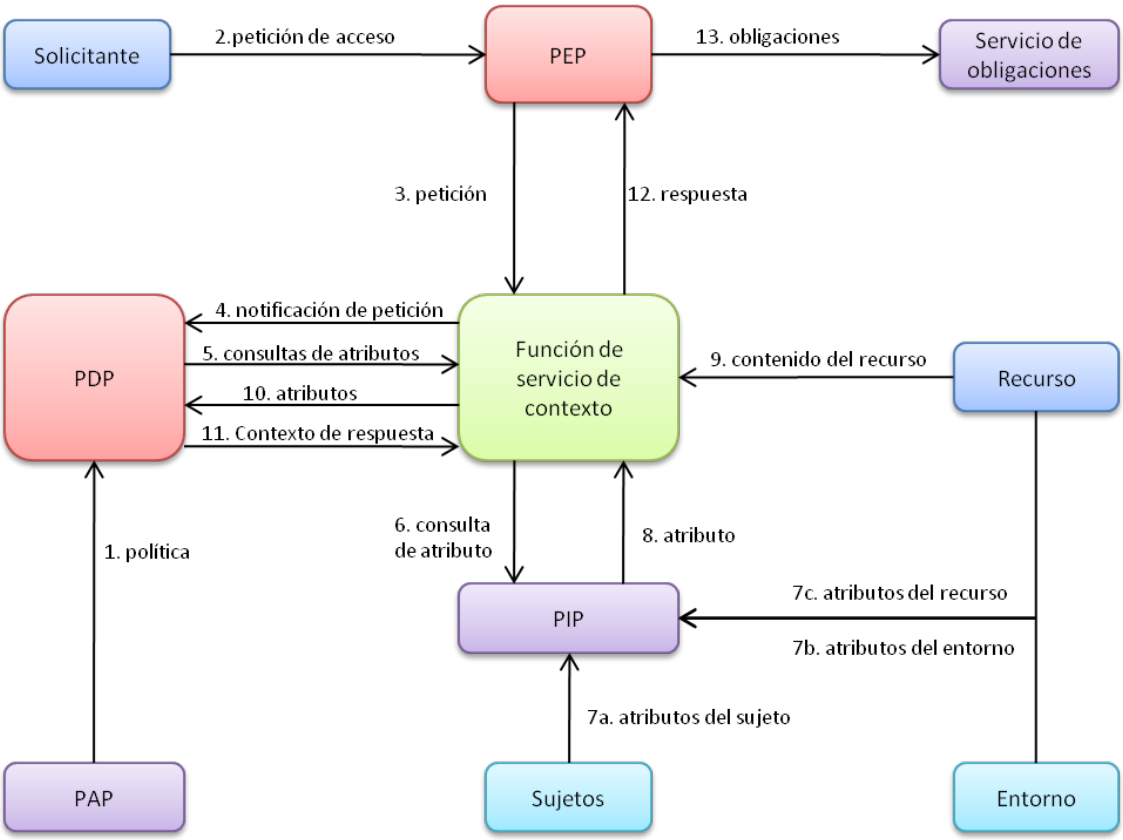

Figura 4.5. Diagrama de flujo de datos del esquema XACML [134]

Pese a la amplia aceptación y las numerosas implementaciones de XACML, el estándar completo presenta una serie de limitaciones [137]:

1. Complejidad: ciertos usuarios de XACML han expresado que su amplio alcance al ser de propósito general y el alto grado de configuración se obtienen a costa de un severo incremento de la complejidad, tanto sintáctica como semánticamente.

2. Rendimiento en tiempo de ejecución: ciertas implementaciones de XACML exhiben degradación en el rendimiento cuando el número de políticas de control de acceso crece.

3. Falta de estructuras jerárquicas de atributos: las actuales versiones de XACML no permiten la organización de atributos de petición como jerarquías “is-a”. Tales jerarquías son útiles para agilizar la especificación de políticas de control de acceso cuando el conjunto potencial de valores de atributos y sus relaciones es extenso.

Finalmente la norma especifica un conjunto de perfiles de aplicación del lenguaje XACML para diferentes escenarios o en conjunción con diversas tecnologías y paradigmas existentes.

1. Perfil de implementación del modelo RBAC con XACML: donde los roles se expresan principalmente como atributos de sujeto XACML. Se puede utilizar un tipo de atributo genérico llamado “rol” que tome valores diferentes para el espectro de roles de la organización (como

"empleado" o "director") o puede haber un tipo de atributo diferente para cada rol ("rol-empleado" o "rol-director") cuyo valor puede estar vacío o contener varios parámetros asociados al rol. Por otro lado XACML soporta múltiples sujetos por petición de acceso indicando varias entidades que pueden participar en dicha petición. En este perfil un atributo de rol puede asociarse a cualquiera de las categorías de sujetos que forman parte de la petición de acceso. Se consideran cuatro tipos de políticas en este perfil:

- <*PolicySet*> **Rol** (*Role* <*PolicySet*> o **RPS**): que asocia a los titulares de un atributo de rol y un valor dados con un <PolicySet> **Permiso** que contiene los permisos propiamente dichos asociados con ese rol, es decir, es una política de asignación rol-privilegios.

- <*PolicySet*> **Permiso** (*Permission* <*PolicySet*> o **PPS**): contiene los permisos propiamente dichos en forma de elementos <*Policy*> y <*Rules*> que describen los recursos y acciones a los que se permite acceder a los sujetos, así como cualquier otra condición acerca de dicho acceso, por ejemplo la hora. Es una política que agrupa privilegios de acceso a recursos y acciones concretas.

- <*Policy*> o <*PolicySet*> **Asignación de roles** (*Role Assignment* <*Policy*> o <*PolicySet*>): define qué roles pueden ser habilitados o asignados a qué sujetos. También puede especificar restricciones sobre combinaciones de roles o números totales de roles asignados o habilitados para un sujeto dado.

- <*Policy*> **TienePrivilegiosDeRol** (*HasPrivilegesOfRole* <*Policy*>): una <*Policy*> opcional de una <*PolicySet*> **Permiso** que soporta peticiones sobre si un sujeto tiene los privilegios asociados a un cometido dado. Si el sistema soporta este tipo de petición, es preciso incluir la <*Policy*> **TienePrivilegiosDeRol** en cada <*PolicySet*> **Permiso**.

2. Perfil de recursos múltiples de XACML: la evaluación de políticas realizada por un PDP se define con un solo recurso solicitado mediante una decisión de autorización que se comunica en un elemento <Result> del contexto de respuesta. Sin embargo en algunos casos puede ser conveniente que un PEP presente un único contexto de petición para acceder a múltiples recursos así como obtener un único contexto de respuesta que contenga una decisión de autorización separada (elemento <Result>) para cada recurso solicitado. Dicho contexto de petición se puede usar para evitar el envío de múltiples mensajes de petición de decisión entre el PEP y el PDP, por ejemplo. Asimismo puede ser conveniente que un PEP presente un único contexto de petición para todos los nodos de una jerarquía y obtener una única decisión de autorización (elemento <Result>) que indique si se permite el acceso a todos los nodos solicitados. Dicho contexto de petición se puede usar cuando el solicitante desea acceder a todo un documento XML, a todo un sub-árbol de elementos en dicho documento, o a todo un directorio de sistemas de archivos con todos sus subdirectorios y archivos, por ejemplo.

3. Perfil de SAML 2.0 de XACML: se detalla en el apartado siguiente tras explicar el protocolo SAML.

4. Perfil de firma digital XML XACML: para la autenticación y la protección de integridad de instancias con objeto de mejorar la seguridad del intercambio de políticas basadas en XACML. La firma digital es útil para la autenticación y protección de integridad sólo si la información firmada incluye una especificación de la identidad del firmante y una especificación del periodo de validez del objeto de datos firmado. En el lenguaje XACML no se ha definido el formato para esta información porque está previsto utilizar otras normas para funciones como la autenticación y la protección de integridad. Se ha definido un formato apropiado en SAML y este perfil recomienda el uso de ejemplares del esquema XACML en Aserciones, Peticiones y Respuestas SAML, las cuales se pueden firmar digitalmente como se especifica en la Recomendación ITU-T X.1141 [138].

5. Perfil de privacidad: la entidad encargada de velar por la seguridad de los datos identificativos tiene la obligación de garantizar que el uso de los mismos está limitado al cumplimiento de los propósitos para los que se registran (u otros propósitos no incompatibles con éstos) así como prevenir la revelación de dichos datos de carácter personal. Esta cláusula proporciona un perfil de atributos normales y un elemento <Rule> normal para cumplir estas obligaciones basados en el propósito para el que se recaba y utiliza la información de tipo personal.

3.3.1. XACML en el dominio sanitario

Una vez que se tiene establecido el mecanismo de control de acceso basado en políticas en lenguaje XACML dentro de una organización es necesario dar un paso más allá en el escenario incluyendo colaboraciones entre organizaciones. Las organizaciones, incluyendo las sanitarias, necesitan un mecanismo para intercambiar sus políticas de seguridad y privacidad, evaluar directivas de consentimiento y determinar autorizaciones de forma interoperable. El perfil de XACML 2.0 para el dominio sanitario XSPA (Cross-Enterprise Security and Privacy Authorization) [139] publicado por OASIS [140] en 2009 proporciona un perfil de seguridad y privacidad entre organizaciones que describe cómo utilizar XACML para proporcionar estas funciones. Define para ello mecanismos para autenticar, administrar y cumplir políticas de autorización controlando el acceso a recursos protegidos residentes en o a través de límites organizacionales.

Como se puede ver en la Figura 4.6, las funciones XACML del PEP son realizadas por la Interfaz de Servicio, y el PEP interactúa con el PIP del Servicio de Atributos y con la funcionalidad de PDP que ahora maneja el Servicio de Control de Acceso (*Access Control Service*, ACS). Las funciones XACML del PAP son realizadas ahora por la Autoridad de Políticas que tiene acceso a las políticas de seguridad controlando el acceso a los recursos protegidos.

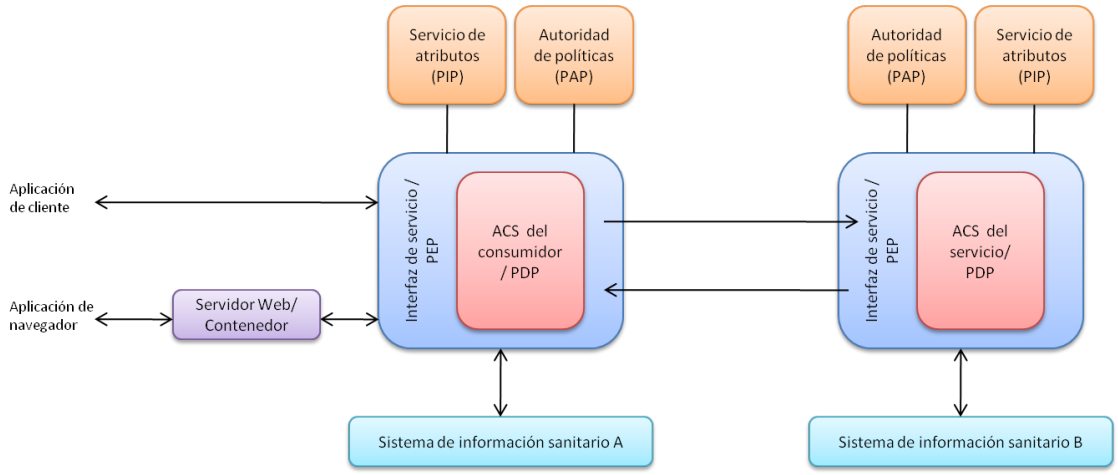

Figura 4.6. Interacción entre componentes del mecanismo de control de acceso [139]

3.4. ITU-T X.1141 - Lenguaje de marcas para la representación de proposiciones de seguridad (SAML)

El lenguaje de marcas para la representación de proposiciones de seguridad (SAML) [138] representa una estructura basada en XML para intercambiar información relativa a la seguridad que se expresa mediante sentencias acerca de sujetos siendo un sujeto una entidad que posee una identidad en algún dominio de seguridad. La norma define además un protocolo que permite que los clientes soliciten sentencias a autoridades y obtengan las respuestas correspondientes. Este protocolo consiste en mensajes de petición y respuesta con formato conforme a un esquema XML. Una proposición constituye un lote de información que suministra cero o varios enunciados expedidos por una *autoridad SAML*. Por lo general, las proposiciones de SAML se expiden acerca de un sujeto representado por el elemento <*Subject*>. No obstante este elemento es opcional y, por consecuencia, otras especificaciones y perfiles pueden utilizar la estructura de aserciones de SAML para emitir enunciados similares sin especificar un sujeto o posiblemente especificando el sujeto de una manera alternativa. Normalmente hay varios proveedores de servicio que pueden aprovechar las proposiciones acerca de un sujeto para controlar el acceso y ofrecer un servicio personalizado. En esta norma se definen tres clases diferentes de enunciados que pueden ser creados por una autoridad SAML (todos están asociados con un sujeto):

- Autenticación: el sujeto de la proposición ha sido autenticado por un medio particular en un momento particular.
- Atributo: el sujeto de la proposición está asociado con los atributos suministrados.
- Decisión de autorización: se ha concedido o denegado una petición de autorización para que el sujeto de la proposición acceda a un recurso específico.

La estructura exterior de una proposición es genérica, es decir, proporciona información que es común a todos los enunciados contenidos en ella. En una proposición, una serie de elementos internos describe la autenticación, el atributo, la decisión de autorización o los enunciados definidos por el usuario que

contienen los datos específicos. Por otro lado, SAML no sólo especifica un lenguaje sino también mensajes de comunicación entre entidades. Así los mensajes de protocolo SAML pueden ser generados e intercambiados gracias a una diversidad de protocolos. La petición de SAML específica y los mensajes de respuesta se deducen de los tipos comunes. El peticionario envía un elemento derivado de *RequestAbstractType* a un respondedor SAML y éste genera un elemento que se adhiere al *StatusResponseType* o se deriva del mismo. En algunos casos, cuando así lo permiten los perfiles, es posible generar una respuesta y enviarla aun cuando el respondedor no haya recibido la petición correspondiente. Los protocolos definidos por SAML pueden realizar las siguientes acciones:

- Devolver una o varias proposiciones solicitadas. Esto puede suceder en respuesta a una petición directa de proposiciones específicas o a una consulta de proposiciones que satisfacen criterios particulares.

- Realizar una autenticación en virtud de una consulta y devolver la proposición correspondiente.

- Registrar un identificador de nombre o dar por terminado un registro de nombre en virtud de una petición.

- Recuperar un mensaje de protocolo que ha sido solicitado mediante algún artefacto.

- Realizar un fin de sesión casi simultáneo de una colección de sesiones relacionadas ("fin de sesión única") en virtud de una petición.

- Proporcionar una correspondencia de identificadores de nombre en virtud de una petición.

3.4.1. Perfil SAML 2.0 de XACML

Este perfil especifica la utilización de SAML 2.0 para proteger, transportar y solicitar instancias de un esquema XACML y otras informaciones necesarias para una implementación XACML [134]. Hay seis tipos de consultas y proposiciones usadas en este perfil (Figura 4.7):

1) *AttributeQuery*: una petición SAML normal utilizada para solicitar uno o varios atributos a una autoridad de atributos.

2) *AttributeStatement*: una declaración SAML normal que contiene uno o varios atributos y se puede usar en una respuesta SAML de una autoridad de atributos o en una aserción SAML como formato para guardar atributos en un registro.

3) *XACMLPolicyQuery*: una extensión de petición SAML para pedir una o varias políticas a un punto de administración de políticas.

4) *XACMLPolicyStatement*: una extensión de declaración SAML que se puede usar en una respuesta SAML de un punto de administración de políticas y también en una aserción SAML como formato para guardar políticas en un registro.

5) *XACMLAuthzDecisionQuery*: una extensión de petición SAML que utiliza el PEP para solicitar una decisión de autorización a un PDP XACML.

6) *XACMLAuthzDecisionStatement*: una extensión de declaración SAML que se puede usar en una respuesta SAML de un PDP XACML o en una aserción SAML que sirve de credencial, pero no es parte del modelo de uso XACML definido actualmente.

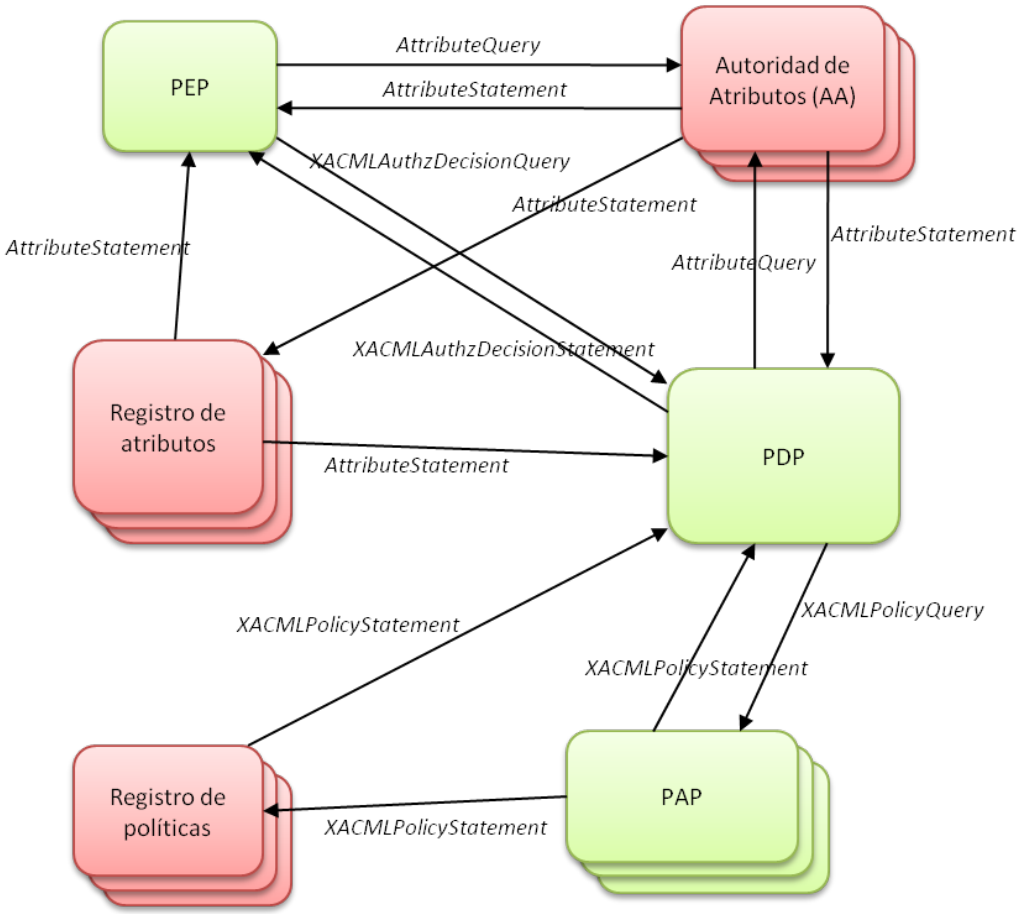

Figura 4.7. Modelo de utilización de XACML con SAML [134]

Este perfil describe todos los elementos del esquema de peticiones y proposiciones, explica su utilización y otros aspectos del uso de SAML con XACML. No se requieren cambios o extensiones a XACML si bien se definen extensiones a SAML. En primer lugar, el esquema de aserciones SAML define una proposición de atributos. El esquema de protocolo SAML define una petición *AttributeQuery* que se utiliza para solicitar ejemplares de proposiciones de atributos y una respuesta que contiene los ejemplares solicitados. Los sistemas que usan XACML pueden usar ejemplares de estos elementos SAML para transmitir y almacenar atributos SAML, y también pueden usar el protocolo *AttributeQuery* SAML

para solicitar ejemplares de atributos SAML. Para poder utilizar un atributo SAML en el contexto Petición XACML es necesario transcribirlo en un atributo XACML.

Por otro lado, SAML 2.0 define una petición de decisión de autorización rudimentaria (*AuthzDecisionQuery*) la cual es incapaz de transmitir toda la información que un PDP XACML puede aceptar como parte de su contexto de petición. Análogamente la declaración de decisión de autorización (*AuthzDecisionStatement*) SAML es incapaz de transmitir toda la información contenida en un contexto de respuesta XACML. Se han definido dos extensiones SAML para permitir que un PEP pueda usar estos mensajes SAML conservando la funcionalidad completa de XACML:

- <*xacml-samlp:XACMLAuthzDecisionQuery*> es una petición SAML que extiende el esquema de protocolo SAML permitiendo que el PEP presente un contexto de Petición XACML en una petición SAML junto con otra información.

- <*xacml-saml:XACMLAuthzDecisionStatement*> es una declaración SAML que extiende el esquema de aserción SAML y permite que un PDP XACML devuelva un contexto de respuesta XACML en la respuesta a una <*XACMLAuthzDecisionStatement*> junto con otra información. También permite que se almacene o transmita un contexto de respuesta XACML en forma de aserción SAML.

Por último, XACML define dos elementos del esquema de políticas: <*Policy*> y <*PolicySet*>. SAML no define protocolos ni esquemas de proposiciones para políticas. Este perfil define nuevas extensiones SAML para los elementos <*XACMLPolicyQuery*> y <*XACMLPolicyStatement*>. Se pueden usar ejemplares de estos nuevos elementos para solicitar, transmitir y almacenar ejemplares XACML <*Policy*> y <*PolicySet*>.

3.4.2. SAML en el dominio sanitario

Al igual que con el caso de XACML OASIS publicó en el año 2009 un perfil de SAML para el dominio sanitario (XSPA, Cross-Enterprise Security and Privacy Authorization) [141] describiendo un marco de trabajo que proporciona interoperatividad de control de acceso a través de organizaciones en el entorno sanitario. La interoperatividad se logra utilizando proposiciones SAML que transportan semánticas y vocabularios comunes. Este perfil define el vocabulario mínimo necesario para proporcionar el control de acceso sobre recursos y funcionalidades dentro y entre sistemas de información sanitarios. En la Figura 4.8 se muestra el diagrama de interacción entre componentes en este escenario inter-organizacional y sanitario.

El servicio de control de acceso (*Access control service*, ACS) recibe la petición del usuario (1) y responde con una proposición SAML que contiene los atributos y autorizaciones del usuario (2). Para realizar esto, el ACS recoge todos los atributos necesarios desde otros recursos o sistemas para crear la aserción solicitada. Por otro lado, este ACS del lado del usuario es responsable de cumplir las políticas de seguridad y privacidad locales. El servicio de control de acceso del lado del proveedor procesa las

proposiciones de petición recibidas (4) y las confronta con las políticas de seguridad y privacidad, tomando una decisión en representación del proveedor del servicio. Entre los atributos que se utilizan en este mecanismo se puede encontrar información relacionada con la localización del usuario, su rol, propósito de uso del recurso, requisitos impuestos por el proveedor del recurso y acciones necesarias para tomar una decisión de control de acceso.

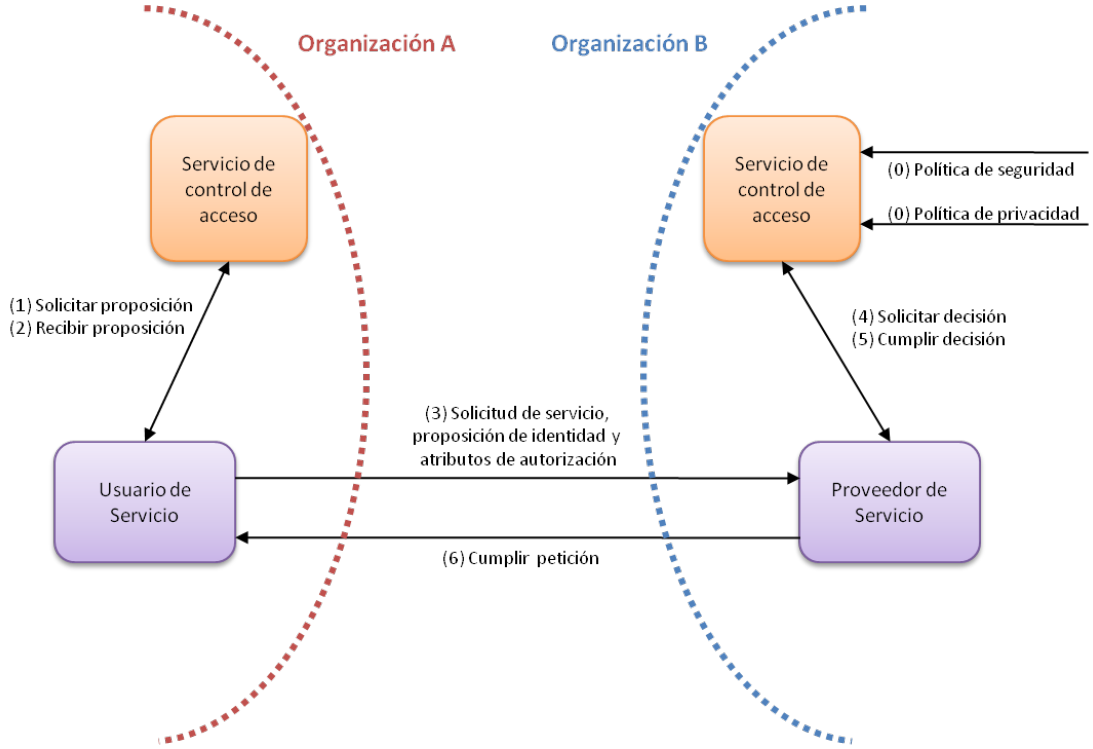

Figura 4.8. Diagrama de interacción entre componentes de SAML XSPA [141]

3.5. Control de acceso del IHE

Dentro del esfuerzo de armonización de estándares en sanidad realizado por el IHE, en el año 2009 publicó un perfil específico de control de acceso [142] dentro del área de infraestructura. Este perfil ha sido desarrollado partiendo de un conjunto de casos de uso que abarcan el espectro de escenarios de control de acceso como son:

- Seguridad interna de recursos (*Internal resource security*): dentro de una organización el acceso a la información médica de un paciente está restringido al personal implicado en el tratamiento del mismo y en las actividades administrativas correspondientes.

- Consentimiento de privacidad del paciente (*Patient privacy consent*): dentro de una red sanitaria regional un paciente puede determinar a qué organizaciones se les permite el acceso a su información sanitaria.

- Uso secundario (*Secondary use*): un paciente puede otorgar derechos de acceso a su información para otros propósitos distintos de su tratamiento en base a unas obligaciones establecidas

previamente (por ejemplo, el paciente puede permitir que se utilice su información médica en un estudio clínico siempre que todos los datos identificativos sean eliminados).

- Situaciones de excepción (*Break glass*): en caso de emergencia, las restricciones de acceso impuestas por las políticas del paciente y las regulaciones internas de seguridad son sustituidas por una política de emergencia que permite a un médico acceder a todos los datos del paciente.

- Opciones individuales (*Individual opt-out*): posibilidad de que un paciente otorgue o deniegue el acceso a individuos concretos (por ejemplo, una enfermera que se opera en el hospital donde trabaja puede desear que empleados de su departamento no tengan acceso a su información médica o administrativa).

De los casos de uso sobre los que se construye este perfil podemos extraer varias consideraciones que deben ser tenidas en cuenta. Por un lado, IHE parte de la base de que un usuario es el dueño de su información y como tal puede elegir quién tiene acceso a ella y quién no, desde organizaciones a individuos concretos. Por otro, esta autoridad solamente puede ser ignorada en situaciones de emergencia que pongan en peligro al propio individuo. Esta filosofía del sujeto asistido como propietario único de su información sanitaria va en consonancia con los futuros planteamientos de los sistemas sanitarios y por ello los esfuerzos de IHE serán fuertemente considerados en el presente trabajo de Tesis Doctoral.

En cuanto a la propia definición del perfil, IHE utiliza como base el esquema de la recomendación XACML si bien se sustituye el servicio de control de acceso (*Access Control Service*) por un sistema de control de acceso (*Access Control System*, ACS). Un ACS es un encapsulado lógico que engloba todos los componentes del subsistema de autorización que están enlazados (lógicamente) con un actor. Cada actor que consume o provee funcionalidad está contenido en un ACS. Además IHE recomienda empezar el desarrollo y despliegue del sistema utilizando roles de la organización para ir extendiendo progresivamente el mismo con otro tipo de atributos que aporten flexibilidad al control de acceso (filosofía ABAC). Además, el consentimiento del paciente está incluido como política/condición para el control de acceso. En la Figura 4.9 se presenta el esquema general de control de acceso de este perfil. En IHE cada actor está asociado a un sistema de control de acceso ACS que está compuesto de los actores lógicos PEP, PIP, PDP y PAP pudiendo cada uno estar distribuido a través de diferentes organizaciones o aparecer en varias instancias (por ejemplo, PIP de recurso y PIP de contexto). Los actores PIP y PAP están representados en la Figura 4.9 por los Servicios de Atributos y las Autoridades de Políticas, respectivamente.

El modelo de control de acceso de IHE, en su caso más simple, representa tres dominios de control de acceso diferentes: de contexto, de sujeto y de recurso. Como puede verse en la figura el Dominio de contexto es aquel en el que se inicia la petición y el cual está sujeto a políticas de control de acceso locales. En este dominio se encuentra el punto de decisión de políticas que resolverá la petición de

acceso. De este dominio se extraerán también atributos del entorno necesarios para la toma de decisiones. Los atributos e identidades del solicitante se encontrarán en Dominios de sujeto. El Dominio del recurso se caracteriza por la gestión de recursos protegidos. A los recursos se les asignan atributos que se consideran metadatos. Para restringir el acceso a los recursos protegidos, políticas de privacidad y seguridad pueden ser evaluadas por un punto de decisión local a este dominio.

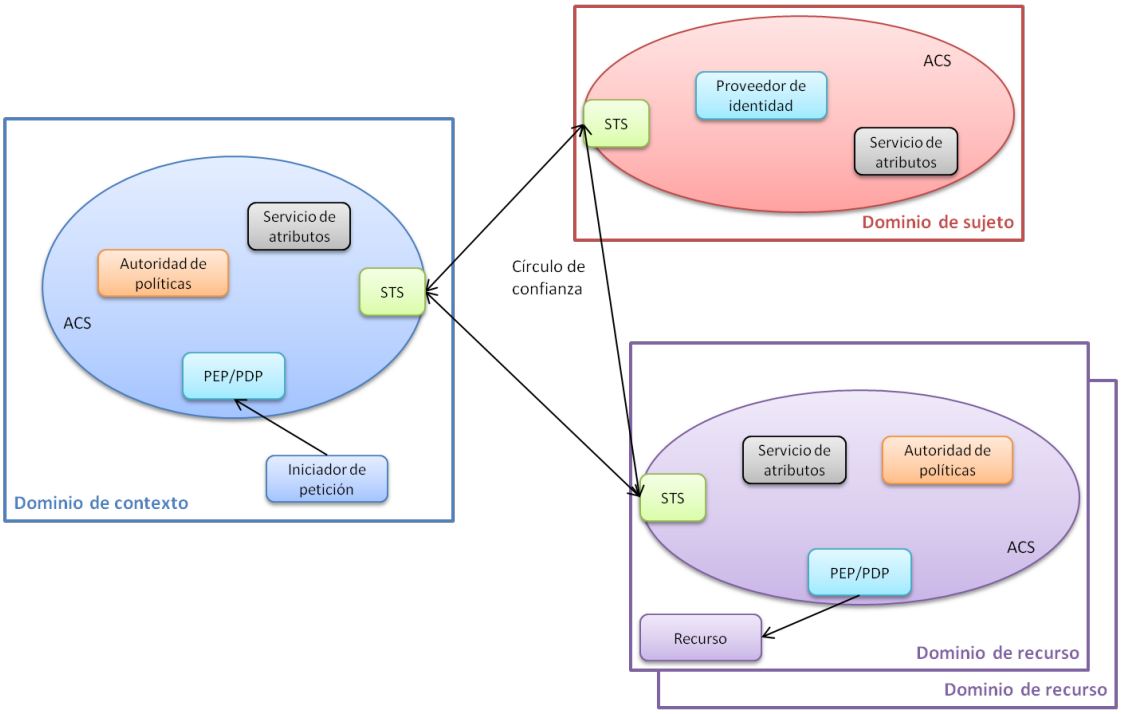

Figura 4.9. Dominios y componentes del control de acceso de IHE [142]

Los diferentes dominios se comunican a través de los STS (*Security token issuing and verification*) elementos encargados del establecimiento de relaciones de confianza entre ACS separados y remotos y para el intercambio seguro (en términos de integridad y autenticidad) de atributos y políticas a través de diferentes dominios. Las Autoridades de Políticas permiten gestionar, seleccionar y activar políticas mientras que los Servicios de Atributos gestionan los atributos de sujetos, recursos, etc., que deben ser considerados para la evaluación de las políticas. Finalmente, los puntos de decisión y cumplimiento de políticas (PEP/PDP) aplican las políticas a las peticiones de servicio.

La mayoría de los elementos del ACS están cubiertos por los perfiles IHE [143]. Por ejemplo, el perfil XUA (*Cross-enterprise user authentication*) el cual describe mecanismos para intercambiar información de sujeto autenticado a través de fronteras administrativas) es el adecuado para conectar el Dominio del sujeto al círculo de confianza. Además se puede utilizar el perfil EUA (*Enterprise user authentication*) como proveedor de identidad basado en Kerberos [144]. El perfil de integración PWP (*Personnel white pages*) define cómo las organizaciones mantienen los atributos de su personal basándose en el protocolo LDAP [145]. Así que, integrando esta información en una proposición XUA o proporcionándola

a usuarios externos a través de "tokens" de seguridad, la funcionalidad requerida por el Dominio del sujeto puede ser proporcionada por los perfiles del IHE. La gestión y provisión de atributos de pacientes está sujeta a los perfiles PIX/PDQ (*Patient identifier cross-referencing/Patient demographics query*). Dependiendo del despliegue, los servicios de atributos pueden estar localizados en el Dominio de contexto, de Recurso u otro dominio dedicado para los pacientes. Por otro lado, el perfil BPPC (*Basic patient privacy consents*) describe cómo los repositorios y registros XDS pueden mantener las políticas de privacidad. Esto permite configurar una autoridad de políticas dentro del Dominio del recurso. A través de un STS, las políticas de privacidad pueden ser intercambiadas de forma segura entre dominios. Los registros XDS están diseñados para la gestión y provisión de atributos de recursos y como tales proporcionan la funcionalidad de los servicios de atributos. Por último, PEP y PDP no están cubiertos por los perfiles de integración IHE.

3.6. Healthcare Information Technology Standards Panel - HITSP

Un esfuerzo similar a la iniciativa IHE es el HITSP [146] que surgió en el año 2005 y cuyo objetivo es establecer un entorno cooperativo entre sectores públicos y privados con el propósito de alcanzar un conjunto de estándares que permitan la interoperatividad efectiva entre aplicaciones software interactuando dentro de las redes de información sanitarias de los EEUU. El marco de armonización de HITSP define un conjunto de artefactos o constructores que especifican cómo integrar y restringir estándares seleccionados para satisfacer los requisitos de los casos de uso y definen un mapa de estándares y armonizan los que se superponen entre sí. Existen diferentes niveles de constructores en cuanto a la amplitud de su alcance y se establecen reglas de composición entre ellos. El nivel más alto es el de las *especificaciones de interoperatividad* que buscan resolver casos de uso específicos. Estas especificaciones se componen de *colaboraciones de servicios* y *paquetes de transacciones* que establecen las relaciones entre servicios o funciones para otorgar funcionalidades avanzadas. En los niveles más bajos se pueden encontrar las *transacciones* que agrupan acciones con funcionalidades complementarias y los *componentes* que reúnen los estándares de base que se utilizan para realizar acciones.

Aunque este marco de armonización pretende cubrir todo el espectro de servicios sanitarios en los que entran en juego requisitos de interoperatividad, el constructor específico del control de acceso es el SC 108 – Colaboración de servicios para el control de acceso (*Access control service collaboration*) [147]. Éste proporciona el mecanismo para establecer autorizaciones de seguridad que controlan el cumplimiento de las políticas de seguridad soportando multitud de esquemas como RBAC, control de acceso basado en entidad, basado en contexto, etc. El SC108 utiliza como precondiciones (o elementos de infraestructura de soporte) los constructores: T17 (Canal de comunicación seguro, *Secured communication channel*) [148] que asegura autenticidad, integridad y confidencialidad de las comunicaciones; y C19 (Aserción de identidad de entidad, *Entity identity assertion*) [149] que permite

recuperar atributos de identidad de la entidad que solicita la decisión de control de acceso. Por otro lado el constructor SC108 utiliza los siguientes, específicos del control de acceso:

- TP 20 – Paquete de transacciones de control de acceso (*Access control transaction package*) [150]: es el núcleo del mecanismo de control de acceso ya que realiza las decisiones; y

- TP 30 – Paquete de transacciones de gestión de directivas de consentimiento (*Manage Consent Directives Transaction Package*) [151]: describe los mensajes que son necesarios para capturar, gestionar y comunicar los derechos que un consumidor otorga o niega a una o más entidades identificadas.

El control de acceso en HITSP cumple con los siguientes requisitos:

1. Las políticas de control de acceso son elementos dinámicos por lo que deben existir capacidades para gestionarlas (creación, modificación, eliminación, suspensión, etc.).

2. El cumplimiento de las políticas de control de acceso es siempre obligatorio.

3. Pueden existir excepciones sobre las políticas pero deben ser estrictamente especificadas y limitadas para evitar que den pie a accesos no autorizados.

4. Los datos de los usuarios pueden estar distribuidos en múltiples sistemas y serán localizados por una entidad con privilegios para realizar dicha actividad. De esta manera ningún atributo de un usuario solicitando acceso se puede quedar fuera del proceso de decisión de control de acceso por no tener derechos para acceder al mismo.

5. Los datos protegidos son accesibles sólo a través de decisiones de control de acceso basadas en el propósito de uso de los mismos y los atributos de información sobre sujetos, recursos, acciones y entorno.

6. Solo usuarios autorizados pueden modificar, actualizar y corregir datos protegidos.

7. Las solicitudes de usuarios de cambios sobre datos protegidos se realizan a los proveedores de esos datos.

8. Antes de otorgar el acceso a datos protegidos puede ser necesario realizar alguna/s acción/es especificadas en términos de obligaciones.

El constructor TP20, núcleo del mecanismo de control de acceso de HITSP, se apoya en los estándares SAML (y perfiles asociados, utilizado como el protocolo de intercambio de atributos de autorización que debe ser soportado), XACML (y perfiles asociados, como mecanismo para expresar políticas y obligaciones de seguridad y privacidad), ASTM International E1986 [152] (seleccionado como

terminología estándar para que los consumidores expresen preferencias que permitan un control de acceso automatizado basado en grupos de personas), OASIS WS-TRUST [153] (método basado en testigos, *tokens*, para solicitar, otorgar, renovar y validar aserciones de autorización de seguridad) y HL7 v3 RBAC [154] (seleccionado para proporcionar el contenido necesario para crear roles interoperables). Además se basa en el estándar ISO 10181-3 como marco de referencia y la Figura 4.10 representa una vista extendida del servicio de control de acceso de esta norma.

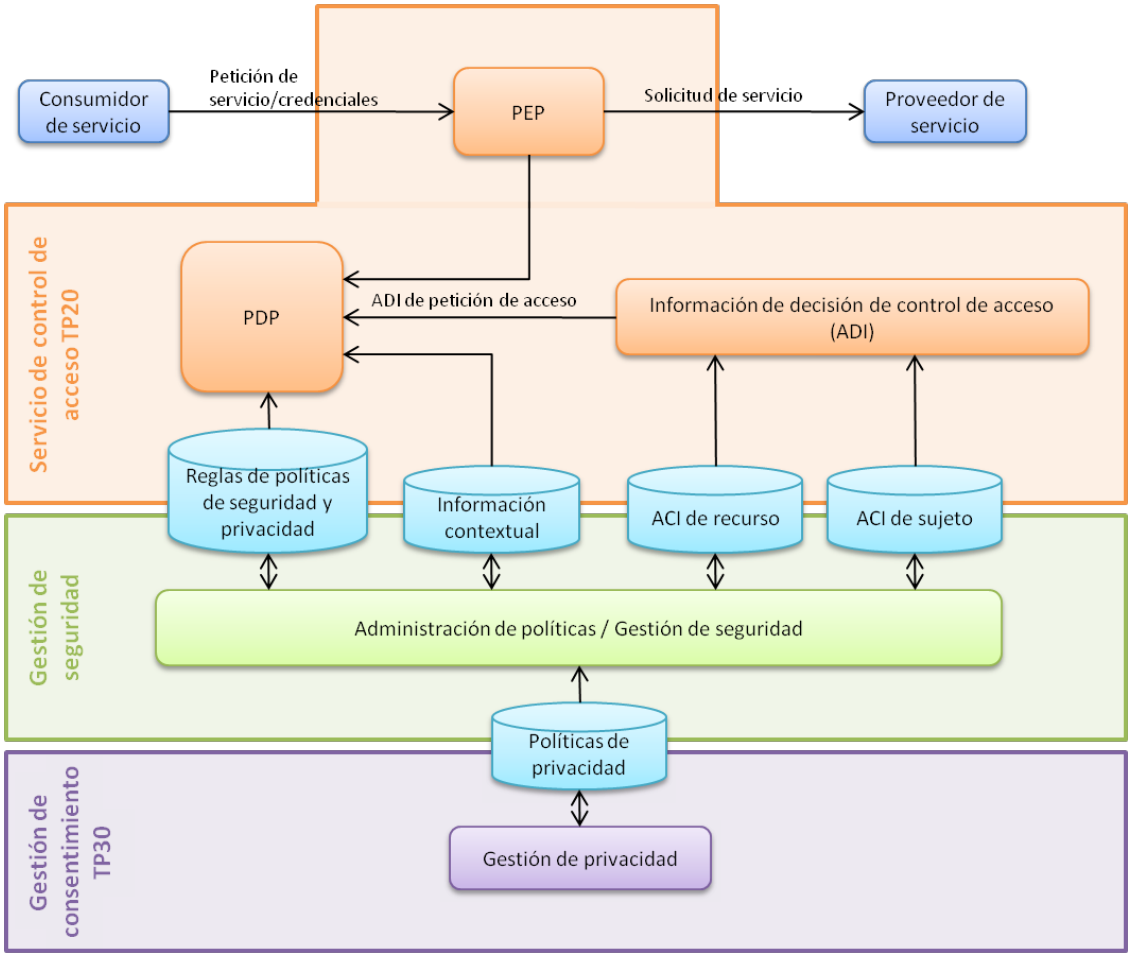

Figura 4.10. Relaciones de componentes en el control de acceso de HITSP [150]

3.7. ISO/TS 21298 – Informática sanitaria – Roles funcionales y estructurales

La especificación de roles para la gestión de privilegios y el control de acceso de RBAC dentro del dominio sanitario está regulada por la norma ISO 21298 [155]. Este documento define una metodología para especificar los roles de un sistema a través de un modelo que los separa en dos grupos principales (los estructurales y los funcionales) y especifica un conjunto básico de roles para el uso internacional de aplicaciones sanitarias. Por un lado los roles estructurales reflejan los aspectos orgánicos de las relaciones entre entidades, esto es, reflejan categorías humanas u organizacionales. Los roles estructurales describen prerrequisitos, competencias y viabilidad para acciones y como están íntimamente relacionados con la estructura de cada organización difieren entre dominios de seguridad separados. Además son relativamente estáticos y su correspondencia con individuos concretos no

cambia en largos periodos de tiempo. Algunos ejemplos de roles estructurales que define la norma son "*Medical Doctor*", "*Dentist*", "*Sanitarian*", "*Child-care worker*", etc. Por otro lado, los roles funcionales están vinculados a la realización de acciones concretas de tal manera que cuando la acción se realiza el rol funcional correspondiente desaparece. Los roles funcionales se pueden definir en relación al proceso de asistencia sanitaria determinando distintos niveles de autorización o derechos de acceso. A diferencia de los estructurales estos roles son altamente dinámicos y algunos ejemplos son "*Subject of care*", "*Administrator*", "*Responsible healthcare professional*", etc. Un conjunto de roles funcionales muy interesante es el relacionado con la información sanitaria especificando quien la crea, modifica, procesa, etc. Este conjunto se compone de "*Composer*", "*Committer*", "*Certifier*", "*Authoriser*", "*Subject of information*" y "*Information provider*".

3.8. Control de acceso en CORBA – Resource Access Decision Facility (RAD)

Como extensión de funcionalidad a la especificación de seguridad definida en CORBA el RAD [156] es un mecanismo para obtener decisiones de autorización y administrar políticas de control de acceso. Proporciona un marco común de solicitud y repuesta de decisiones de autorización independiente de las aplicaciones, de hecho esta funcionalidad está encapsulada como un elemento externo a las mismas. Es necesario puntualizar que aunque el mecanismo RAD está especialmente desarrollado para un entorno CORBA conserva mucha independencia de la tecnología y, por ello, es un componente sofisticado muy valioso desde un punto de vista teórico (especialmente por su origen en el ámbito sanitario y su filosofía de distribución de componentes con software de intermediación) y por eso ha sido incorporado en esta revisión del estado del arte de control de acceso.

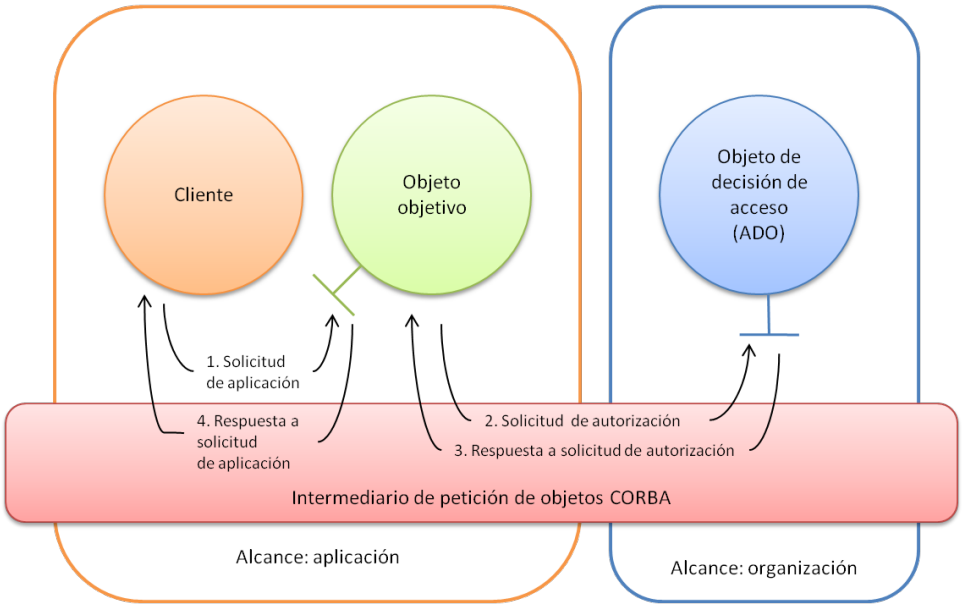

Figura 4.11. Esquema de flujo de mensajes de autorización [156]

En la Figura 4.11 se representa el esquema del proceso de petición y resolución de una decisión de control de acceso. Un cliente invoca una operación de la interfaz proporcionada por el objeto objetivo y,

mientras se procesa, este objeto solicita una decisión de autorización al objeto de decisión de acceso (*Access decision object*, ADO). Éste consulta otros objetos internos al RAD para resolver la decisión que es devuelta al objeto objetivo como booleano. Este objeto es entonces el responsable de cumplir la decisión. La Figura 4.12 muestra el modelo de decisión de acceso en el que se apoyan los modelos de información y computacional del RAD.

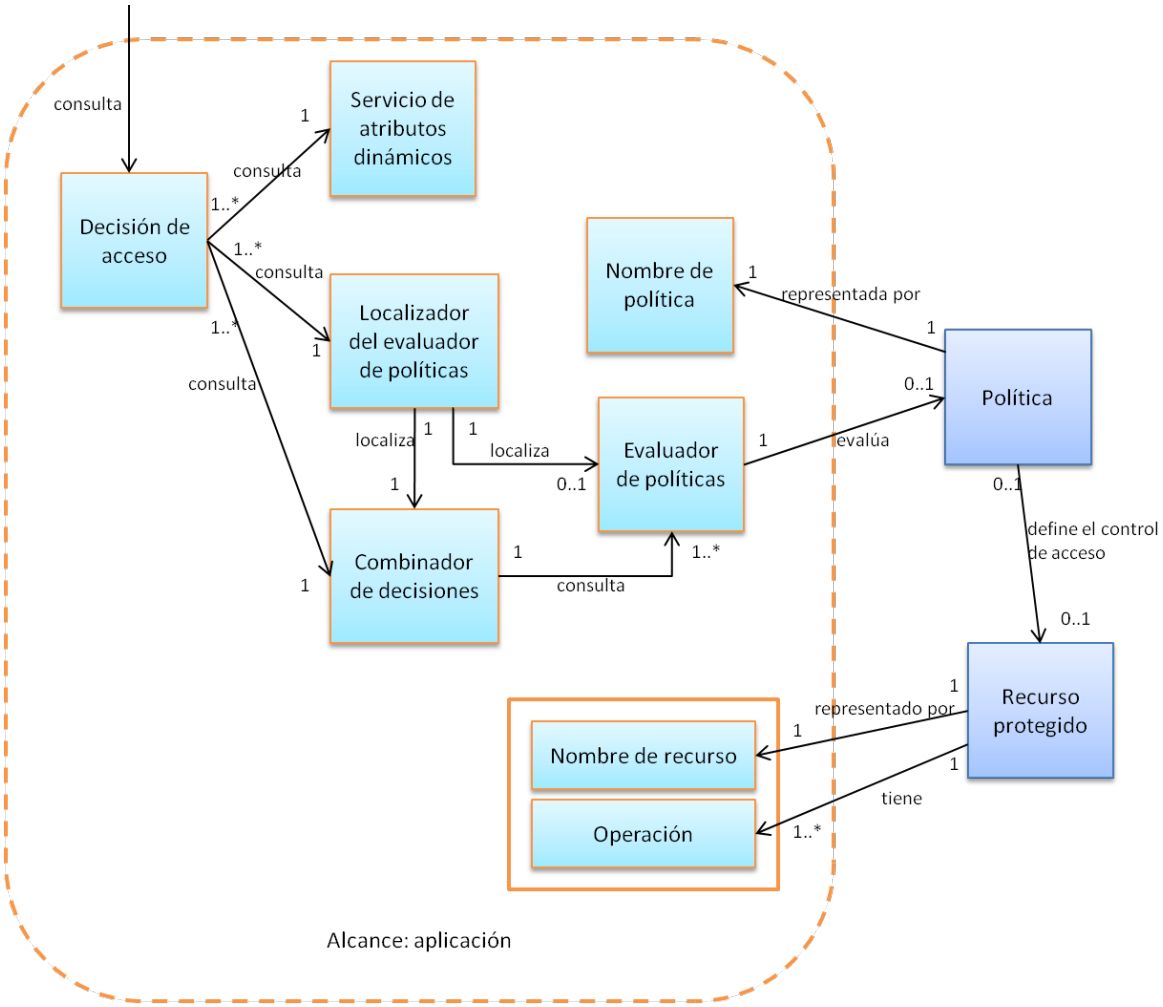

Figura 4.12. Modelo de control de acceso del OMG RAD [156]

3.9. ISO/TS 22600 – Informática sanitaria – Gestión de privilegios y control de acceso

La norma ISO/TS 22600 [157-159] introduce principios y especifica servicios necesarios para la gestión de privilegios y el control de acceso en sistemas de información sanitarios poniendo especial interés en las colaboraciones entre organizaciones a través de fronteras administrativas y de seguridad. Esta norma se divide en tres partes y se basa en un modelo conceptual donde los servidores de autorización locales y los servicios de repositorios de políticas y directorios de identidades pueden asistir al control de acceso. El acceso estará basado en cuatro puntos: (1) la identificación autenticada del usuario; (2) las reglas de acceso correspondientes al recurso específico; (3) las reglas que recogen los atributos de autorización que el gestor de autorizaciones ha proporcionado al usuario; y (4) las funciones de la aplicación específica.

3.9.1. Parte 1 – Visión general y gestión de políticas

La primera parte de la norma [157] se centra en la colaboración entre gestores de autorización que pueden operar sobre fronteras organizacionales con políticas de seguridad separadas. La colaboración es definida en un acuerdo de políticas firmado por todas las organizaciones implicadas y constituye la plataforma básica para la operación. Esta norma es más de carácter administrativo que técnico proponiendo un formato de documento como base del acuerdo de colaboración al margen de detalles de implementación o específicos de plataforma. No especifica tampoco servicios o protocolos de seguridad de comunicación ni técnicas de autenticación. Por otro lado define los objetivos y la estructura de la gestión de privilegios y el control de acceso. Sus objetivos son:

a) presentar indicaciones para la compartición de información a través del acuerdo de colaboración,

b) ser un estándar para el control de acceso y la gestión de privilegios que gobiernan el intercambio seguro de información entre dominios de seguridad separados, y

c) establecer una ruta para la transformación de los sistemas existentes para que sean conformes con los criterios de esta parte para el intercambio de información transfronterizo.

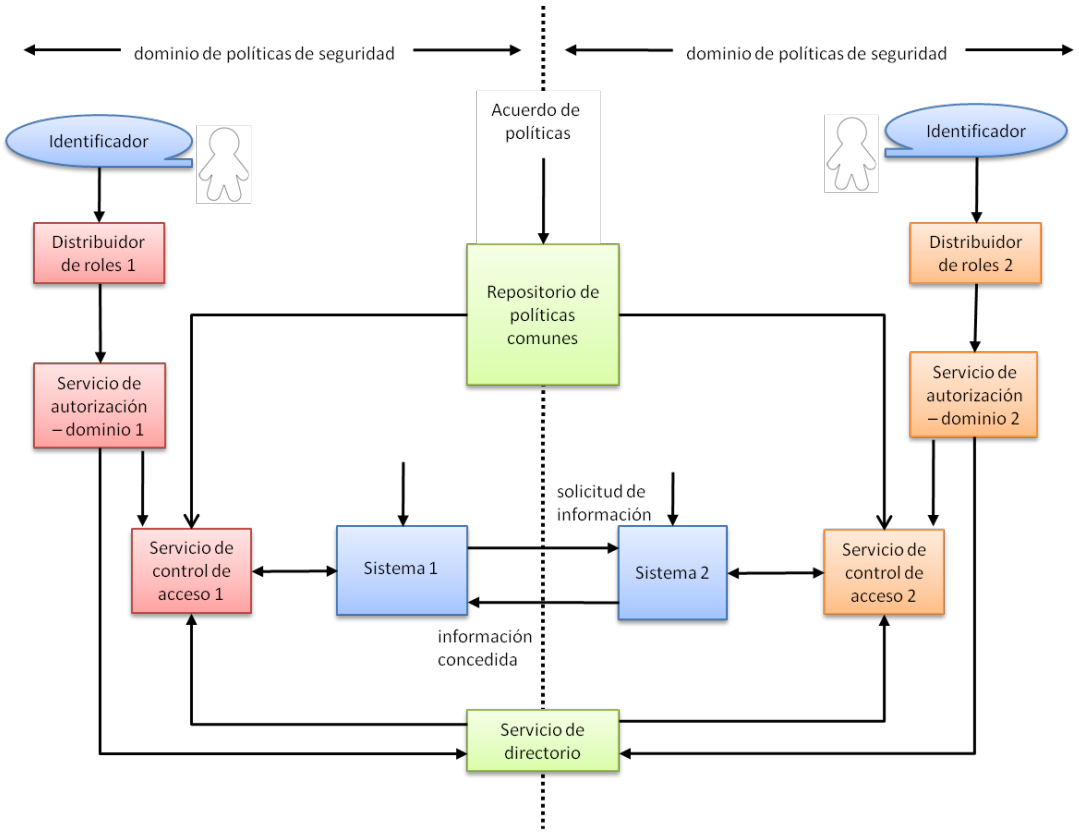

Figura 4.13. Modelo de seguridad de la norma ISO22600 [157]

La estructura de gestión de privilegios consiste en: dominios, políticas (de control de acceso y acuerdos de colaboración), roles, repositorio de políticas, directorio, mecanismos de autenticación y proceso de intercambio de información. En la Figura 4.13 se muestra la relación entre estos conceptos.

3.9.2. Parte 2 – Modelos formales

Esta parte [158] introduce los modelos formales de alto nivel para los componentes de la estructura de gestión de privilegios y control de acceso basados en el marco de referencia ODP. Para los distintos modelos se utilizan esfuerzos y estándares previos y se enfoca una posible arquitectura de sistemas de información sanitarios a partir del modelo de componentes genéricos GCM. Los modelos que se presentan son:

- Modelo de dominio: un dominio de seguridad es caracterizado por un identificador, un nombre, una autoridad de dominio y un calificador. Estos datos (Tabla 4.IV) siguen el estándar HL7 v3 Definición de tipos de datos [99]. La comunicación entre dominios se lleva a cabo a través de conjuntos de políticas comunes que describen el marco legal de operación y colaboración.

Atributo	Nombre	Tipo
Identificador de dominio	*domain_identifier*	*SET <OID>*
Nombre de dominio	*domain_name*	*BAG<EN>*
Identificador de la autoridad	*domain_authority_ID*	*OID*
Nombre de la autoridad	*domain_authority_name*	*ST*
Calificador de dominio	*domain_qualifier*	*CS*

Tabla 4.IV. Atributos del dominio de políticas de seguridad [158]

Atributo	Nombre	Tipo
Identificador de política	*policy_identifier*	*SET <II>*
Nombre de política	*policy_name*	*CS*
Identificador de la autoridad	*policy_authority_ID*	*OID*
Nombre de la autoridad	*policy_authority_name*	*ST*
Identificador de dominio	*domain_identifier*	*SET <OID>*
Nombre de dominio	*domain_name*	*BAG<EN>*
Lista de objetivos	*target_list*	*LIST<INT>*
Identificador de objetivo	*target_ID*	*SET<II>*
Nombre de objetivo	*target_name*	*EN*
Objeto objetivo	*target_object*	*II*
Código de operación	*operation_code*	*CE*
Políticas	*policies*	*CD*

Tabla 4.V. Atributos de política básica de seguridad [158]

- Modelo de documento: procesos, entidades, roles, etc., deben estar documentados y aceptados por las partes implicadas. Se debe soportar la transferencia de firmas entre organizaciones a través de sintaxis de mensajes encriptados.

- Modelo de políticas: una política es la formulación del concepto de requisitos y condiciones para la creación, almacenamiento, procesado y uso fiables de información sensible. Es necesario destacar que la norma sólo se refiere a información y no a recursos en general como objetos a proteger. Se especifican un conjunto de campos que debe tener una política (también tomando como base el documento de definición de tipos de datos de HL7 v3) que se muestran en la Tabla 4.V. Por otro lado se define la clase política y el conjunto de subclases que la especializan (ver Figura 4.14).

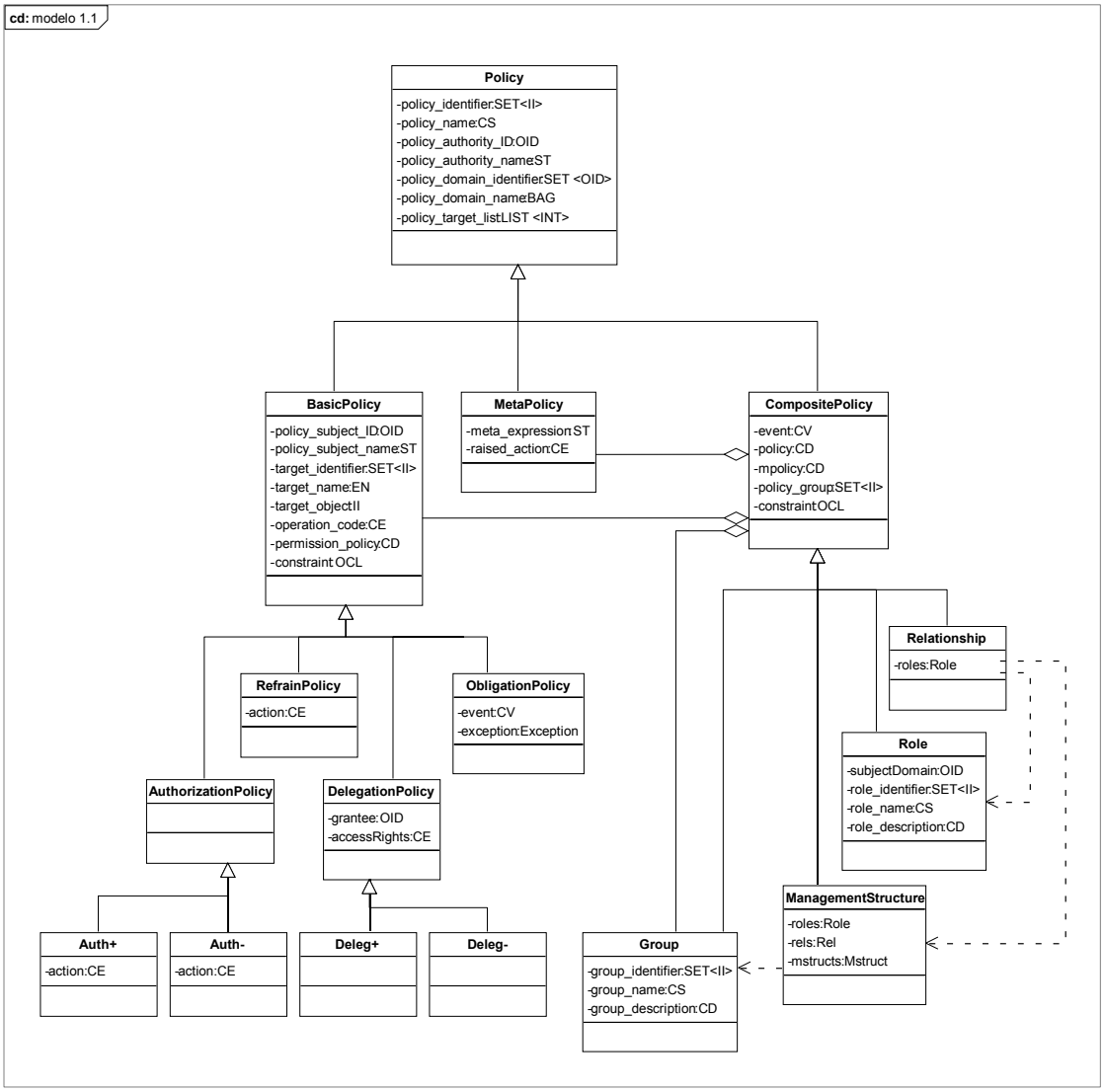

Figura 4.14. Diagrama de clases de políticas

El modelo de políticas define cuatro tipos básicos: de autorización (*authorization*, que definen las acciones permitidas), de obligación (*obligation*, que son disparadas por eventos y definen las acciones que tienen que ser realizadas por los agentes gestores), de abstención (*refrain*, definiendo

acciones que los sujetos deben eludir) y de delegación (*delegation*, que determinan qué autorizaciones pueden ser delegadas a quién). Los tipos compuestos son: grupos (*groups*, definiendo el alcance de políticas relaciones a las cuales se les aplica el mismo conjunto de restricciones), roles (*roles*, que definen grupos de políticas) y de relaciones (*relationships*, que describen grupos de políticas relacionadas con la interacción entre conjuntos de roles).

- Modelo de roles: este modelo sigue la especificación de roles de la norma ISO/TS 21298 y que los agrupa en dos categorías: funcionales y estructurales.

- Modelo de autorización y asignación de privilegios y roles: la concesión de credenciales, privilegios y las tareas de autorización son realizadas conectando roles con políticas. A los individuos se les otorgan certificados de asignación de roles. Por otro lado, los privilegios se asignan a un rol a través de certificados de especificación de roles en lugar de a individuos concretos. Los certificados de asignación de roles pueden ser certificados de atributos o de clave pública pero los de especificación de roles sólo pueden ser del primer tipo.

- Modelo de control: ilustra cómo el control es ejercido sobre el acceso a una operación de objeto sensible. Se definen cuatro componentes en el modelo: el solicitante (*claimant*), el verificador (*verifier*), el objetivo (*target*) y la política de control (*control policy*). En la Figura 4.15 pueden verse las relaciones entre ellos.

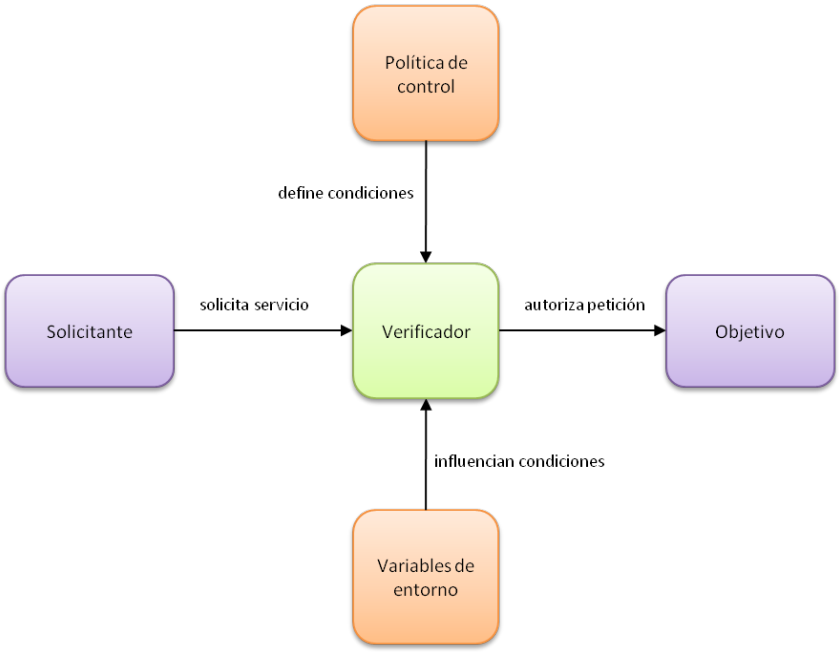

Figura 4.15. Modelo de control de la norma ISO/TS 22600

- Modelo de delegación: como extensión al modelo de control es necesario un modelo de delegación de privilegios. En la Figura 4.16 se muestran los componentes principales: verificador, fuente de autoridad (*source of authority*) y solicitante. El verificador dota con privilegios globales a

una entidad conocida como fuente de autoridad la cual es una autoridad de atributos que delega privilegios a los solicitantes a través de certificados de atributos. Opcionalmente un solicitante puede delegar sus privilegios o parte de ellos a otro solicitante. Como caso general, el verificador debe confirmar que todas las entidades en el camino de delegación poseen suficientes privilegios para acceder al objetivo elegido por el solicitante directo.

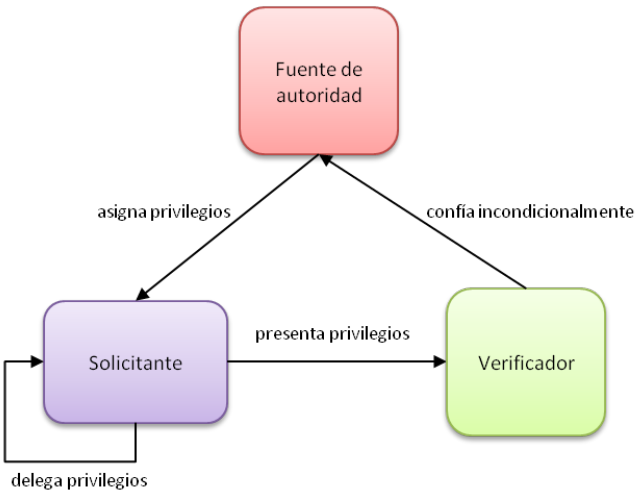

Figura 4.16. Modelo de delegación de la norma ISO/TS 22600

- Modelo de control de acceso: los modelos anteriores se combinan en el modelo de alto nivel completo de control de acceso. Los elementos principales son las entidades involucradas, los roles, los permisos, las operaciones y los objetos. La gestión del control de acceso está caracterizada por los siguientes componentes: definición de roles y restricción de los mismos; asignación usuario-rol; asignación rol-permiso; y asignación de restricciones para activación de roles asignados al usuario. En esta norma se presenta un esquema de control de acceso basado en roles (Figura 4.17) basado en los modelos anteriores y los estándares RBAC.

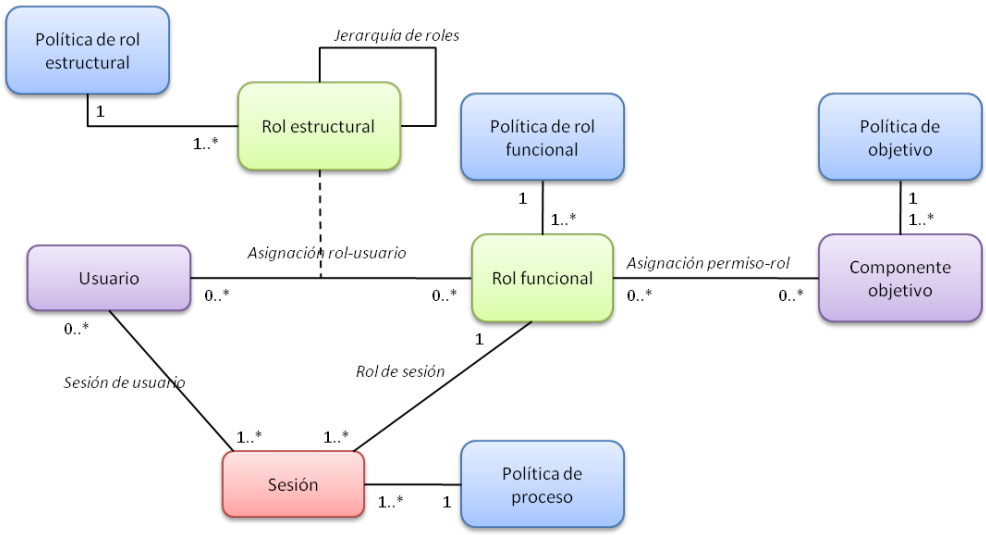

Figura 4.17. Esquema de control de acceso basado en roles conducido por políticas

3.9.3. Parte 3 – Visión general y gestión de políticas

La tercera parte de la norma [159] introduce ejemplos para especializar e implementar los modelos formales de alto nivel definidos en la segunda parte en componentes arquitecturales basados en el marco de referencia ODP. Las especificaciones se derivan de los estándares XML, SAML y XACML y han sido armonizadas con partes esenciales de la infraestructura de gestión de privilegios definida en la norma ASTM E2505-07 [160]. En este punto del presente trabajo de Tesis Doctoral buscamos la armonización de estándares de control de acceso y gestión de privilegios independientes de plataformas o tecnologías concretas. Por ello no se profundiza en esta parte de la norma puesto que presenta un conjunto de alternativas tecnológicas en este campo. En la implementación que se llevará a cabo como parte de los resultados se tendrán en cuenta los aportes de esta parte de la norma ISO/TS 22600.

3.10. ISO/DTS 14441 – Informática sanitaria – Requisitos de privacidad y seguridad para pruebas de conformidad de sistemas de HCE

Actualmente existe en desarrollo en ISO una especificación técnica centrada en los requisitos de privacidad y seguridad que debe cumplir cualquier sistema que maneje registros de historia clínica. La especificación ISO/DTS 14441 [161] revisa el conjunto de requisitos que debe satisfacer este tipo de sistemas para todas las disciplinas de seguridad. Se destacan a continuación los requisitos de aquellas áreas que resultan de especial interés para el presente trabajo: el consentimiento del sujeto de la asistencia y el control de acceso. Por un lado, las organizaciones sanitarias necesitan conocer que han obtenido el consentimiento requerido en su jurisdicción cuando recogen, usan o revelan información sanitaria personal. Por ello, los principales requisitos que se deben satisfacer con respecto al consentimiento del sujeto de la asistencia son:

- Registro de consentimiento: deben existir mecanismos para registrar el consentimiento del sujeto de la asistencia incluyendo la posibilidad de ocultarlo o revocarlo; además, el consentimiento puede ser para toda la información del sujeto, para sólo una parte o para un propósito específico.

- Registro mínimo de datos: junto al consentimiento debe registrarse la fecha en que fue creado así como el tipo de consentimiento cuando pudiesen existir varios (por ejemplo, consentimiento explícito frente al implícito).

- Directrices junto a los datos: se deben proporcionar mecanismos para transmitir las restricciones sobre la revelación de los datos junto a los propios datos si el receptor de los mismos no pudiera de otra forma informarse de las directrices marcadas por el sujeto de la asistencia sobre la revelación de sus datos.

- Acceso en emergencias: en situaciones especiales permitidas por la legislación o políticas locales puede ser necesario acceder a los registros de los pacientes sin considerar las directivas de consentimiento previamente registradas. Esta capacidad de acceso en emergencias sólo será

proporcionada a usuarios autorizados y su invocación (así como la razón por la que se está obviando el consentimiento del sujeto de asistencia) será almacenada en un registro de auditoría.

- Consentimiento dado por un representante legalmente autorizado: cuando sea posible que un representante legalmente autorizado pueda establecer directrices de consentimiento sobre los datos de un sujeto de asistencia, la identidad de este representante y su relación con el sujeto debe ser registrada junto a las directrices.

- Informe sobre cambios de consentimiento: debe ser posible indicar qué directrices de consentimiento estaban vigentes en cualquier momento dado para cualquier sujeto de asistencia.

Los requisitos específicos para el control de acceso son:

- Controles de acceso: los sistemas deberán verificar que cada persona o entidad autenticada que solicita acceso a información sanitaria personal está autorizada para acceder a dicha información.

- Control de acceso basado en roles: los sistemas deberán soportar RBAC capaz de mapear cada usuario a uno o más roles y cada rol a una o más funciones del sistema o privilegios de acceso.

- Otras formas de control de acceso: los sistemas deberían adicionalmente ser capaces de mapear cada usuario a derechos de acceso asignados o restringidos basados en grupos de trabajo a los que el usuario pertenece o el contexto de la transacción.

- Delegación de acceso a la información sanitaria personal de sujetos de asistencia: los sistemas deben ser capaces de mantener una asociación entre usuarios y los registros de sujetos de asistencia y permitir el acceso en base a esta asociación. Así un usuario con acceso autorizado a los registros de un sujeto de asistencia podrá otorgar derechos de acceso a esos registros a otro usuario.

- Informar de privilegios de acceso: los sistemas deberían ser capaces de informar, para un usuario dado, si puede acceder a los registros de un sujeto de asistencia concreto y los privilegios que el usuario tiene sobre los registros del sujeto.

- Restricciones en privilegios de acceso: cuando a un usuario le han asignado más de un rol los sistemas permitirán que el usuario seleccione qué rol desea que se aplique a su sesión de usuario.

- Revocar privilegios de acceso: los sistemas soportarán la revocación de los privilegios de acceso de un usuario sin requerir la eliminación del usuario del sistema.

3.11. HSSP – Servicios de seguridad, privacidad y acceso

El proyecto HSSP tiene como objetivo la identificación y documentación de especificaciones de servicio, funcionalidades y soporte a la conformidad relevantes para los participantes dentro del ámbito sanitario

y con especial esfuerzo en implementaciones en el mundo real. Dentro de esta iniciativa se encuentra el grupo de trabajo PASS (*Privacy, Access and Security Services*) [162] cuyo trabajo aún está en proceso (por ello la información de este apartado podría cambiar en el futuro cuando se publiquen las especificaciones definitivas). Actualmente se cuenta con una especificación del modelo conceptual del control de acceso en HSSP [163] que describe los puntos de vista a nivel conceptual asociados con los requisitos de negocio que están relacionados con el contenido, estructura y comportamiento funcional de la información importante para el control de acceso en entornos sanitarios. Utiliza cuatro puntos de vista:

- de negocio (*business viewpoint*): identifica el contexto y alcance de la especificación y contiene casos de uso y escenarios, un conjunto de requisitos, y las entidades y objetos implicados incluyendo los roles que desempeñan estas entidades. Se basa en los modelos formales de la norma ISO/TS 22600 y los esquemas de control de acceso de la ISO 10181-3 y HITSP (Figura 4.10) conectando la capa de la Figura 4.18 con la de gestión de consentimiento. Los roles identificados son el PDP, el PEP, el proveedor de políticas (*PolicyProvider*) y el proveedor de información de decisión de acceso (*AccessDecisionInformationProvider*).

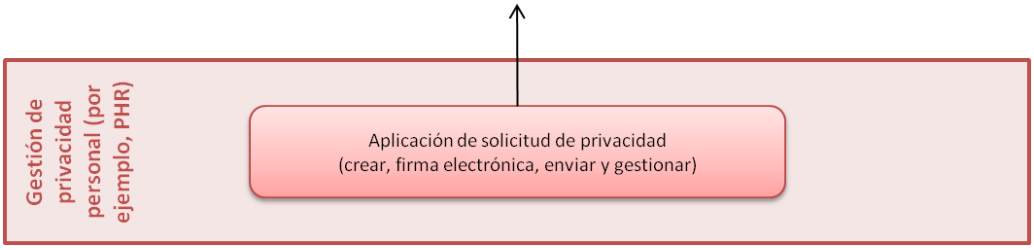

Figura 4.18. Extensión al esquema de control de acceso de la Figura 4.10

- de información (*information viewpoint*): presenta el modelo de objetos representando los requisitos de información del sistema, esto es, las entidades principales y sus relaciones. Este punto de vista incluye un modelo estático (conteniendo los objetos de información y esquema invariante) y un modelo dinámico (que identifica cambios de estado permitidos a los objetos de información).

- computacional (*computational viewpoint*): recoge el comportamiento funcional de un servicio de control de acceso agrupado de tal forma que pueda estar distribuido y expuesto a través de interfaces de servicio. Las capacidades identificadas son: 'cumplir decisión de control de acceso', 'solicitar decisión de control de acceso', 'obtener información de decisión de acceso', 'obtener política', 'enviar política de control de acceso', 'enviar registro de auditoría' y 'gestionar política'.

- de ingeniería (*engineering viewpoint*): identifica y captura cualquier capacidad relevante para la plataforma y documenta los requisitos esenciales para la distribución de la funcionalidad identificada.

4. Tecnologías semánticas al servicio del control de acceso

Las tecnologías de gestión semántica están alcanzando una gran repercusión debido al potencial subyacente en su filosofía (automatización de tareas, promoción de la interoperatividad entre sistemas, revelación de relaciones implícitas en conjuntos de datos, etc.) y esto hace que gradualmente estén siendo aplicadas a todas las disciplinas de los sistemas de información, no siendo una excepción la disciplina de control de acceso. Estas tecnologías operan a dos niveles: uno conceptual y abstracto en el que se codifican las semánticas (significado de los conceptos, relaciones, etc.) y otro de aplicación. En los entornos que utilizan semántica es necesario tener especificación de políticas de autorización e inferencia en ambos niveles, así se podría definir una regla de permiso sobre un concepto abstracto de recurso (por ejemplo, "artículo") y que esa regla se propague para todos los recursos que instancia el concepto ("artículo_23" o "artículo_de_conferencia_17"). A continuación se exponen algunas de las iniciativas más relevantes en la aplicación de tecnologías semánticas al control de acceso y la gestión de privilegios. Se podrá comprobar que son posibles diferentes aproximaciones al problema en función del aspecto (o aspectos) que se aborden con semántica. En un intento de organizar estas propuestas se ha optado por separar aquellas centradas en el nivel conceptual abstracto de las que trabajan en el nivel de aplicación o individual.

4.1. Control de acceso: nivel conceptual abstracto

4.1.1. CLAC - Control de acceso a nivel conceptual (*Concept-Level Access Control*)

Especialmente desarrollado para la Web Semántica, CLAC [164] es un modelo de control de acceso capaz de especificar autorizaciones sobre conceptos definidos en ontologías y llevándolas a cabo sobre instancias de datos anotadas por esos conceptos. Una característica del control de acceso a documentos XML (base de la Web Semántica) es que permite especificar cómo el acceso a un elemento puede ser propagado a otros que estén estructuralmente relacionados con el primero. Las propagaciones están basadas en las relaciones semánticas entre conceptos u ontologías. El modelo se basa (casi en exclusiva) en un lenguaje de control de acceso semántico (*Semantic Access Control Language*, SACL) para expresar las autorizaciones de acceso a nivel de concepto. El SACL utiliza la sintaxis y vocabulario de OWL introduciendo nuevo vocabulario específico para el control de acceso. El modelo CLAC se centra en la utilización de dicho lenguaje para expresar políticas y la anotación de documentos XML con conceptos recogidos en una ontología. Una de las debilidades de esta iniciativa es que no posee una semántica clara para sus reglas de inferencia de políticas.

4.1.2. SAC – Control de acceso semántico (*Semantic Access Control*)

El modelo SAC [165] está basado en las propiedades semánticas de los recursos que tienen que ser controlados, las de los clientes que solicitan el acceso a los mismos, características del contexto y de los certificados de atributos creados por el sistema de control de acceso. Una de las novedades del modelo SAC es que aborda los problemas de distribución y multi-granularidad de las políticas, así el control de

acceso a los recursos es independiente de la localización y se ha desarrollado contando con diferentes sistemas de control de acceso y entidades de autorización trabajando simultáneamente. SAC ha sido implementado sobre un lenguaje llamado Lenguaje de Políticas Semánticas (*Semantic Policy Language*, SPL) que utiliza las propiedades semánticas de los recursos y el contexto para especificar los criterios de control de acceso y la integración semántica de entidades de autorización externas. SPL se fundamenta en cuatro elementos:

- metamodelo de políticas (*policy metamodel*): representa las semánticas de los criterios de control de acceso,

- metamodelo de especificación de aplicabilidad de políticas (*Policy Applicability Specification*, PAS): representa la semántica sobre la localización de las políticas de los recursos, es utilizado para localizar la política correcta para cada recurso basándose en las propiedades relevantes del recurso,

- metamodelo de representación de recursos protegidos (*Secured Resource Representation*, SRR): representa la semántica sobre los recursos, y

- metamodelo de descripción de fuentes de autorización (*Source of Authorization Description*, SOAD): incluye la semántica de los diferentes atributos que pueden ser certificados por una entidad de autorización.

La principal desventaja del modelo SAC es que no considera el efecto de las relaciones semánticas en la propagación de políticas.

4.1.3. SBAC – Control de acceso semántico (*Semantic-Based Access Control*)

Un modelo más completo que CLAC es SBAC [166] ya que mientras CLAC trabaja con ontologías para los objetos, SBAC considera relaciones susceptibles de inferencia en todos los dominios de control de acceso, es decir, añade los dominios de sujetos y acciones. La semántica conceptual de SBAC es una extensión de la semántica de OWL extendida con la de SWRL. Define un conjunto de ontologías que cubren todo el dominio del control de acceso:

- ontología de seguridad basada en semántica: recoge la semántica (conceptos y relaciones) de las políticas de control de acceso;

- ontología SBAC: incluye los sujetos, objetos y acciones; y

- ontologías de privilegios y de prohibiciones: extienden la ontología SBAC con permisos y prohibiciones concretas entre sujetos, objetos y acciones.

Algunas implementaciones de elementos del modelo SBAC han sido desarrolladas a través de herramientas semánticas (Protégé, Jena y otros) y para su aplicación a la Web Semántica se integró la ontología SBAC con OWL-S. SBAC es similar a SAC en motivación y trasfondo, sin embargo el seguimiento de OWL y la pila de estándares de la Web Semántica es más escrupuloso en SBAC. Por el contrario, la semántica de SAC está definida como XML Schema heredando las limitaciones inherentes a XML. Como desventaja, SBAC no profundiza en la semántica de las reglas de inferencia de políticas, dejando un aspecto esencial sin explorar.

Existe otra propuesta con el mismo nombre, SBAC [167] que también utiliza OWL y SWRL. Este modelo contiene una ontología base (compuesta de sujetos, objetos y acciones), otra de autorización y una más de operaciones administrativas (otorgar o revocar derechos de acceso). Esta última ontología es una de las diferencias más patentes entre ambas iniciativas, ya que el primer modelo descrito no contempla esta posibilidad. Aparte de este aspecto, el resto de características de ambas propuestas son bastante similares.

4.2. Control de acceso: nivel individual

4.2.1. Rei

Rei [168] es un lenguaje de especificación de políticas propuesto para la web semántica y basado en conceptos lógicos utilizando RDFS los que permiten especificar entidades y propiedades de ellas, acciones, políticas y actos. Las políticas son descritas en términos de permisos (lo que un agente puede hacer), prohibiciones (lo que no puede hacer), obligaciones (lo que debería hacer) y dispensaciones (lo que un agente ya no necesita hacer). Por otro lado, los actos son: delegación (añade un permiso), revocación (elimina un permiso o añade una prohibición), solicitud (causa que una acción sea realizada) y cancelación (cancelar una petición previa). Aunque el lenguaje fue originariamente orientado a la especificación de políticas para sujetos, objetos y acciones individuales (reales) también permite la especificación de políticas basadas en roles, grupos o entidades aunque no estén explícitamente declarados en la ontología Rei. Debido a que el lenguaje Rei se centra en el nivel individual, es decir, describe entidades y acciones identificables en el mundo real, no puede aplicar un completo razonamiento e inferencia sobre los conceptos a nivel abstracto perdiendo una de las ventajas del uso de semánticas para el control de acceso.

4.2.2. Servicios de gestión de dominio y políticas KAoS

KAoS [169] es uno de los primeros esfuerzos para representar políticas utilizando un lenguaje de la web semántica, en este caso OWL. Los servicios y herramientas KAoS permiten la especificación, gestión, resolución de conflictos y cumplimiento de políticas en contextos específicos establecidos por organizaciones representadas como dominios. El uso de KAoS está principalmente orientado a las tecnologías de agentes, sin embargo su aplicación en servicios web semánticos como servicios grid o

servicios de agentes está extendida también. El núcleo de esta propuesta es la ontología de políticas KAoS (*KAoS Policy Ontology*, KPO) que define los conceptos utilizados para describir el entorno y las políticas en un contexto genérico. KPO es una ontología de base en la que se apoyan otras, extendiendo los conceptos, con nociones específicas del entorno controlado particular y dominio de aplicación. Las políticas se pueden basar en autorizaciones y obligaciones positivas o negativas, además de otras acciones como delegación. Como le ocurre a Rei, KAoS trabaja con instancias del nivel individual y pierde funcionalidad al no poder ser eficientemente aplicado a nivel conceptual.

4.2.3. XACML

Aunque XACML es un lenguaje para la especificación de políticas no emplea relaciones semánticas entre los conceptos. Pese a ello, el marco de trabajo XACML ha servido de base para introducir ontologías y tecnologías semánticas en el modelo de control de acceso. No es la intención de este apartado repasar todas y cada una de las iniciativas en este ámbito puesto que son numerosas y variadas [170-173]. Además, debido a que estos esfuerzos no se pueden considerar como normalizados, sólo deben servir como una muestra de implementaciones prácticas en las que nos basaremos en el capítulo siguiente.

Sección 3. Resultados

Capítulo 5. Resultados

1. Introducción

El objetivo principal del presente trabajo de Tesis Doctoral es contribuir al diseño de arquitecturas sanitarias distribuidas que potencien la interoperatividad entre sistemas y soporten escenarios de asistencia sanitaria centrados en el ciudadano. Este objetivo se concreta en la que es la principal contribución original de este trabajo: la definición, diseño y despliegue de un paradigma de asistencia sanitaria centrado en el sujeto de la asistencia que se ha venido a denominar POVO (*Person-Oriented Virtual Organization*). El escenario POVO se sustenta sobre los fundamentos de interoperatividad, apertura y estandarización, y para desarrollarlo se han llevado a cabo un conjunto de contribuciones en diversos campos como la formalización de arquitecturas de servicios, la estandarización en mecanismos de autorización o la gestión de semántica. En la Figura 5.1 se recogen los resultados que han dado lugar al paradigma POVO y que han sido obtenidos siguiendo un enfoque metodológico de aplicación de TIC en el dominio sanitario.

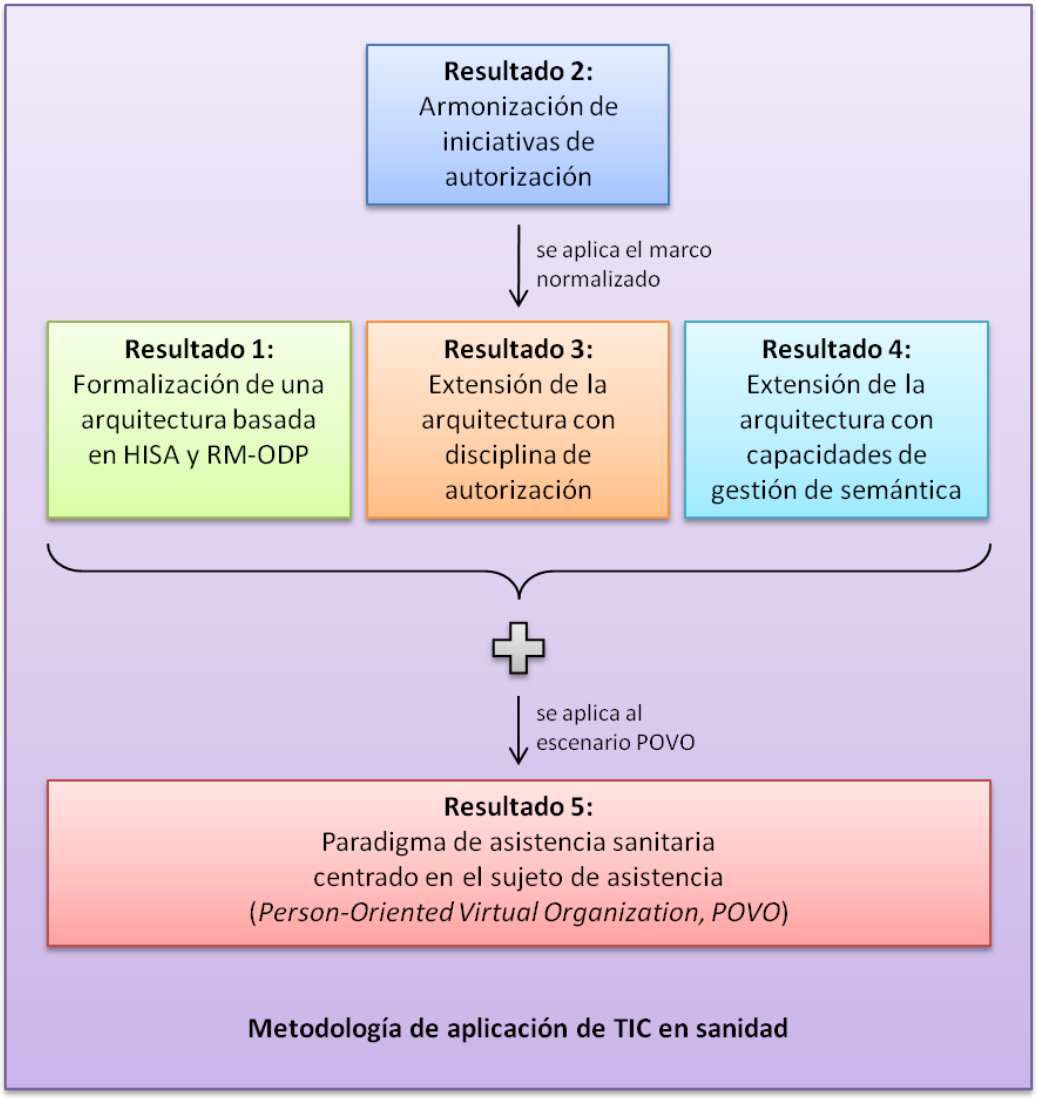

Figura 5.1. Resultados obtenidos durante el desarrollo de la Tesis Doctoral

Si bien la figura anterior resume los resultados obtenidos a alto nivel, es en el esquema de la Figura 5.2 donde se muestra en detalle cómo los diferentes resultados permiten desarrollar el escenario POVO. En primer lugar se han seleccionado las normas HISA y RM-ODP como base para la formalización de la arquitectura de soporte siguiendo los resultados de la comparativa de marcos de trabajo realizada en el Capítulo 2. Esta formalización permite disponer de una arquitectura básica de referencia sobre la que desarrollar las capacidades y componentes necesarios para la POVO facilitando la interoperatividad y apertura del sistema final. HISA y RM-ODP acotan el conjunto de herramientas de formalización y, en este caso, se ha hecho uso de la norma ISO 19793 (UML4ODP) para especificar la arquitectura de referencia. Además se ha reestructurado la norma HISA para mejorar su conformidad con el modelo de referencia ODP y así optimizar su aplicabilidad como estándar de desarrollo de arquitecturas de sistemas distribuidos en el dominio sanitario. Esta contribución es la que se ha venido a llamar *Resultado 1* en la Figura 5.1.

La arquitectura de referencia basada en las normas HISA y RM-ODP cubre sólo aspectos básicos de los sistemas distribuidos en el dominio sanitario y, por tanto, el paradigma POVO ha precisado desarrollar en mayor detalle varias características de la arquitectura. En concreto se han abordado y extendido dos aspectos: los mecanismos de autorización y las capacidades de gestión de semántica. Por un lado la disciplina de autorización cuenta con numerosas iniciativas y estándares así que, buscando mantener la normalización de la arquitectura, se ha definido un marco de armonización entre dichas propuestas. Este esquema armonizado (*Resultado 2*) ha sido formalizado dentro de la arquitectura de referencia mediante el perfil UML4ODP y es el que da pie al control de acceso a los recursos en el escenario POVO (*Resultado 3*). Por otro lado, se han extendido las capacidades transversales de la arquitectura en materia de gestión de elementos semánticos (ontologías, motores de inferencia, etc.) siguiendo también las directrices de formalización de la norma UML4ODP (*Resultado 4*).

Para la implementación del paradigma POVO se particulariza la arquitectura de los resultados anteriores para el estilo arquitectural SOA y las tecnologías Grid (*Resultado 5*). Una de las características más originales del escenario POVO es que está centrado en el sujeto de la asistencia y éste es el administrador del acceso a sus recursos con capacidad para otorgar y revocar privilegios de acceso. Por ello se ha implementado con mayor detalle el mecanismo de autorización de la POVO basado en el esquema normalizado de control de acceso e incorporándole capacidades de gestión de semántica. Los diferentes componentes que forman parte del mecanismo han sido implementados mediante diversas técnicas y tecnologías semánticas (ontologías, reglas, motores de inferencia, lenguajes específicos de dominio…) buscando obtener un sistema final orientado a cualquier tipo de usuario ocultando la complejidad inherente.

El desarrollo aquí explicado así como la definición y fundamentos del concepto POVO se exponen con detalle en los apartados siguientes.

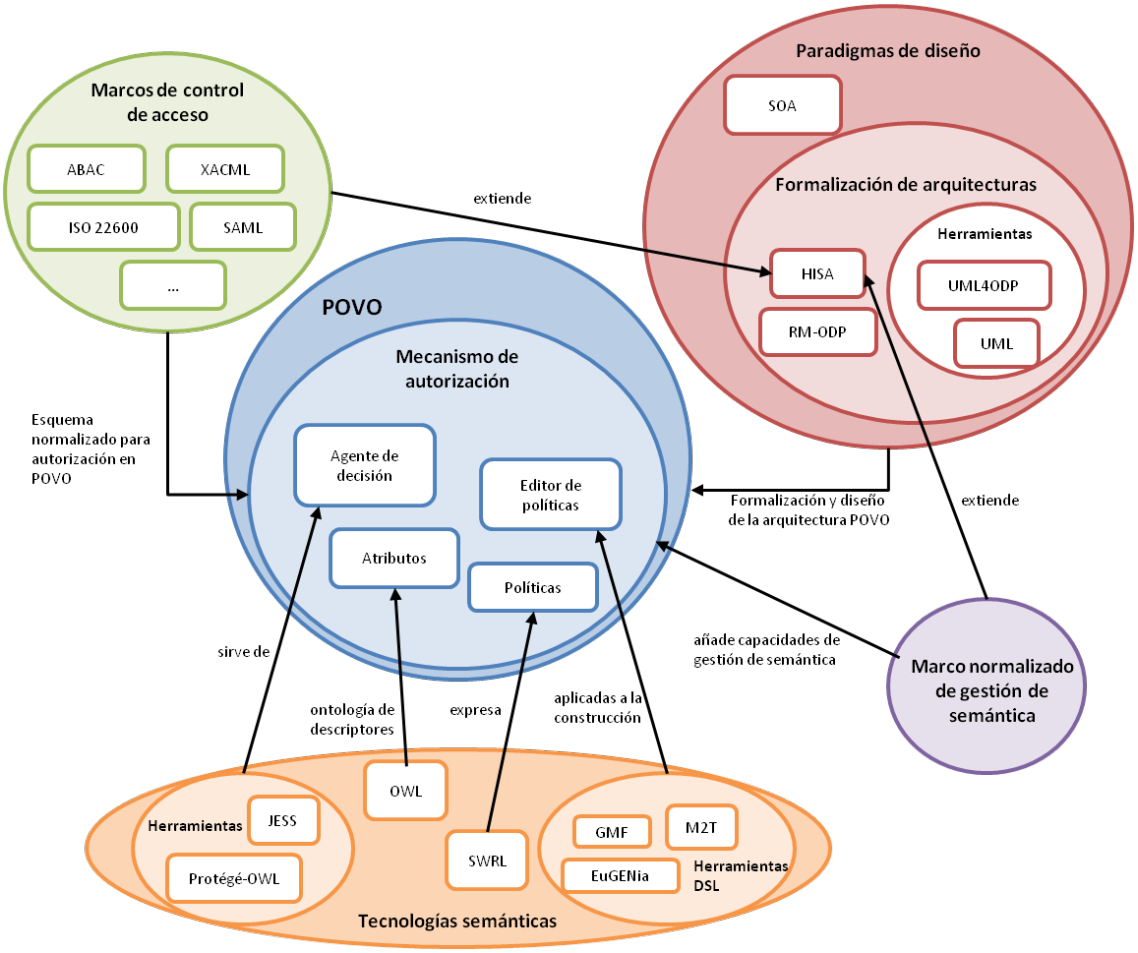

Figura 5.2. Relación entre las contribuciones originales de la Tesis Doctoral

1.1. Metodología de aplicación de las TICs en sanidad

Un importante obstáculo al que se enfrentan los arquitectos y diseñadores de sistemas en el dominio sanitario es la fragmentación de las soluciones y la falta de interoperatividad entre sistemas. Este problema supone un escollo para la innovación tecnológica del dominio sanitario y la mejora de la relación coste-efectividad de las propuestas desarrolladas. Debido a esta heterogeneidad de dispositivos y sistemas aislados, los esfuerzos de los últimos años han estado más centrados en la evaluación de los sistemas TIC de asistencia que en el desarrollo de marcos metodológicos para la correcta aplicación de estas tecnologías al dominio sanitario [174]. La aplicación de las TICs en sanidad de forma metodológica será un factor clave para resolver los nuevos escenarios y obstáculos, reduciendo los costes asociados y mejorando la eficiencia de los procesos sanitarios. Fruto del análisis de metodologías de propósito general se plantea un método de aplicación de TICs en el dominio sanitario.

La Figura 5.3 muestra las diversas fases de esta metodología (adoptada en el presente trabajo de Tesis Doctoral) la cual parte de varios aspectos considerados fundamentales para el éxito de las soluciones desarrolladas y que permiten garantizar su usabilidad y evolución a lo largo del tiempo a la vez que

minimizan las posibilidades de quedar obsoletas cuando avance la tecnología o surjan nuevos sistemas. Resumimos a continuación algunos de estos aspectos clave.

1. El usuario en el centro del proceso de diseño y desarrollo

Uno de los factores que más afectan al éxito de las aplicaciones TIC en sanidad es el entendimiento imperfecto entre las partes implicadas en el diseño, desarrollo y uso de los sistemas, especialmente cuando se establecen los requisitos de los usuarios. Cualquier sistema de información necesita poner al usuario en el centro del diseño. Esto se aplica con mayor énfasis en el dominio sanitario y principalmente cuando los sistemas están directamente implicados en la asistencia sanitaria de los pacientes pudiendo ser éstos usuarios del sistema (aparte de los usuarios profesionales). Reconocer a los usuarios finales como partes claves a involucrar en todas las etapas del diseño y desarrollo de los sistemas puede contribuir a que éstos tengan mayores oportunidades de éxito una vez desplegados y en uso.

2. Conceptos abiertos para el diseño

Desde un punto de vista metodológico la definición de las necesidades y requisitos de los sistemas debe realizarse atendiendo a dos conceptos fundamentales. Por un lado, el proceso de diseño no se aplica a un diseño único sino a un "espacio de diseño" en el que un diseño particular debe concebirse teniendo en cuenta un posible diseño alternativo [175] en función de las necesidades del contexto y escenario de aplicación, lo cual implica la consideración, desde el comienzo del diseño, de las posibles modificaciones derivadas de las diferentes necesidades específicas que pueden tener los ciudadanos. El segundo concepto es el de "diseño para todos" [176] que se refiere al diseño de aplicaciones, servicios y productos interactivos utilizables en los distintos escenarios bajo consideración incorporando la filosofía de espacio de diseño. Esta metodología facilitará la escalabilidad del sistema y la posibilidad para realizar posibles cambios de escenario de los pacientes al sólo tener que agregar o eliminar servicios.

3. Integración mediante una arquitectura abierta y estandarizada

Una de las carencias más importantes de los desarrollos de sistemas TIC aplicados al dominio sanitario es que las resoluciones tecnológicas presentan diseños cerrados para cada contexto socio-sanitario y escenario físico de aplicación. De manera que un usuario que solicite una cartera de servicios médicos y/o sociales de diversa naturaleza se enfrenta a la paradoja de tener que emplear varios sistemas de atención remota simultáneamente en función de cada una de sus necesidades. El inconveniente principal de esta disociación de la atención al usuario en diversas soluciones aisladas es su difícil integración al haber sido concebidas de manera independiente en su etapa de diseño. Para la integración de sistemas se precisa de una arquitectura tecnológica distribuida, abierta y escalable que proporcione servicios que puedan cubrir las necesidades propias de cada contexto médico y escenario de aplicación.

4. La evaluación continua en el proceso de desarrollo

A menudo la etapa final de desarrollo de un sistema comprende su evaluación y es en ese momento donde aparecen fallos y errores que difícilmente son subsanables en una fase tan avanzada. Por ello la evaluación debe asumirse desde el principio del proyecto como una parte integral del mismo. Esta filosofía es incorporada, por ejemplo, por el procedimiento de desarrollo Harmony [115] lo que ayuda a la medición de la seguridad del sistema ya que cada uno de los componentes hardware y software del sistema se evalúan independientemente antes de su integración. También se mejora la robustez del sistema al ir adquiriendo conocimiento y experiencia de los posibles fallos y de cómo solucionarlos lo que permite una especial atención y previsión de alternativas adecuadas cuando la avería pueda tener consecuencias sobre la asistencia al paciente.

5. Separación del diseño de los sistemas en aspectos dependientes e independientes de tecnología

Las especificaciones de diseño que centran aspectos independientes de tecnología de un sistema pueden ser reutilizadas con distintas implementaciones tecnológicas sin más que desarrollar los aspectos dependientes para la tecnología concreta. Esta descripción independiente de tecnologías permite enfrentar un importante obstáculo que sufren los sistemas de información en general, la portabilidad. Contar con un diseño independiente de tecnología permite reutilizar esfuerzos pasados y sólo centrarse en aquellos aspectos dependientes de la tecnología y que evolucionan con ésta.

6. Reutilización de sistemas legados

Uno de los fundamentos que deben sustentar una metodología orientada al dominio sanitario (con todas las características y problemas reseñados) es la reutilización de sistemas legados que adquieren nuevas funcionalidades a través de la promoción de características como apertura, interoperatividad, flexibilidad, etc.; y su integración con los nuevos sistemas.

En la metodología utilizada (Figura 5.3) se consideran dos casos diferentes. Por un lado (el más común en los escenarios sanitarios actuales), está en el que se cuenta con sistemas legados, esto es, componentes que ya realizan alguna actividad aunque de manera parcial o totalmente aislada. Se procede entonces a considerarlos parte del nuevo sistema y a reutilizar sus funcionalidades, por ejemplo, a través de interceptores. A menudo este caso es subestimado o ignorado por los nuevos desarrollos considerando que siempre la mejor solución es la que parte de cero y desechando lo que existía previamente. Puede que la solución que surge de un desarrollo desde cero sea más eficiente pero en ese caso no se tienen en cuenta los costes que se invirtieron en los sistemas ya desplegados y que, aun con funcionalidad reducida, aún están en funcionamiento. Para estos casos se propone la herencia de estos sistemas y reutilización de sus capacidades pero siempre exigiéndoles (al igual que se hace con los nuevos desarrollos) que cumplan con los requisitos de apertura, estandarización e interoperatividad. El segundo caso obviamente es aquel que trata los desarrollos de sistemas completamente nuevos que no heredan ningún componente de anteriores soluciones.

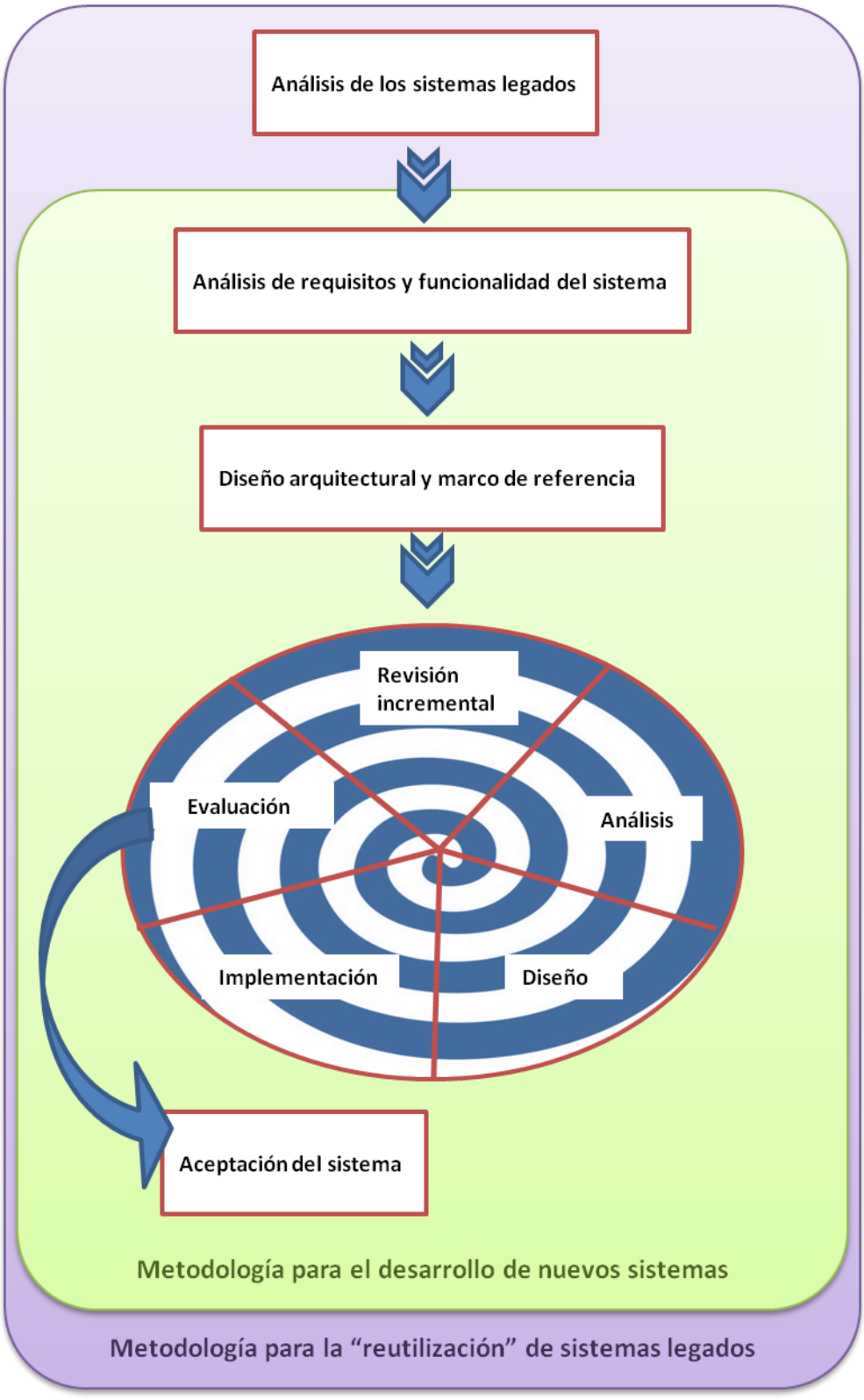

Figura 5.3. Metodología para el desarrollo de sistemas TIC en el dominio sanitario

La metodología está dividida en dos ciclos: uno primero en cascada para la identificación de requisitos y el marco arquitectural y otro iterativo en el que se diseña, implementa y evalúa el sistema en un proceso incremental. A continuación se describen en detalle cada una de estas fases y qué actividades

se desarrollan en ellas. Como esta metodología se ha utilizado en el desarrollo del trabajo presentado en este capítulo, en cada fase se mencionarán aquellos puntos del mismo que corresponden con las actividades descritas.

Fase 1. Análisis de los sistemas legados

Esta es la primera fase para el caso en el que se considere la reutilización de componentes ya desplegados y cuya funcionalidad quiera heredarse en el nuevo sistema. En primer lugar es necesario analizar estos elementos y discriminar su funcionalidad de la tecnología de implementación. Esas capacidades serán utilizadas en el diseño del nuevo sistema aunque no se debe olvidar que están limitadas a una tecnología concreta. Desechar esta restricción se asemeja a crear un nuevo sistema desde cero y no es lo que se considera en este caso. En una etapa posterior, y una vez que se diseñe el sistema completo, se deberán analizar las posibles vías de integración de los componentes legados con el nuevo sistema, teniendo en cuenta la posible heterogeneidad de tecnologías. Además de la funcionalidad de estos elementos, se deben extraer los conceptos utilizados y la semántica de los mismos. El registro de estos conceptos puede realizarse de varias maneras aunque el uso de ontologías locales puede facilitar la futura integración con los conceptos del sistema completo a través del mapeo de ontologías. Asimismo se debe identificar y documentar el contexto de cada uno de los componentes legados, es decir, los actores que interactúan con ellos, los recursos utilizados, funcionalidades de la plataforma que los soportan, etc. El trabajo desarrollado no hereda sistemas ya desplegados luego esta fase no ha sido necesario ejecutarla.

Fase 2. Análisis de requisitos y funcionalidad del sistema

Tanto si estamos en un escenario con componentes legados como si no, en esta fase se realiza un análisis del mismo para identificar las necesidades y poder así especificar los requisitos a satisfacer por el nuevo sistema. Se define la funcionalidad y objetivo del sistema a alto nivel y se identifican los actores y procesos que estarán afectados por el nuevo sistema. Paralelamente se realizará una consideración de los requisitos tecnológicos y de negocio impuestos por el entorno en el que se desplegará el sistema. El detalle de procesos, interacciones, actores, etc., se realizará una vez que se entre en la zona iterativa del proceso metodológico. Las actividades realizadas en esta fase se corresponden con los análisis llevados a cabo para la especificación de los requisitos del paradigma POVO (ver Apartado 6.2) en general y la funcionalidad del mecanismo de control de acceso y el elemento de gestión de semántica (Apartados 4.2.1 y 5.2).

Fase 3. Diseño arquitectural y marco de referencia

En esta metodología se apuesta por las soluciones efectivas, interoperables y abiertas, con capacidad para evolucionar en el tiempo y no quedar obsoletas ante nuevas funcionalidades, sistemas o tecnologías. Por ello un aspecto fundamental es que los sistemas desarrollados sean conformes con los

estándares y marcos normalizados de cada una de las disciplinas que cubran (comunicación, seguridad, almacenamiento de datos…). Para ello esta fase cubre el análisis e identificación de todos los estándares y recomendaciones que deben ser adoptados por el sistema a desarrollar. En numerosas ocasiones será necesario identificar lagunas o superposiciones entre estándares y resolverlas. Una vez finalizada esta fase se dispondrá de un marco de referencia compuesto de estándares cubriendo todas las disciplinas necesarias que guiarán el proceso de desarrollo del sistema hasta alcanzar una solución conforme a todos los estándares considerados. Poniendo como ejemplo las contribuciones del presente trabajo de Tesis Doctoral, esta fase correspondería con la armonización de estándares y recomendaciones de seguridad que se lleva a cabo en el Apartado 3. Dentro de la definición del marco normalizado se identificará la arquitectura de referencia sobre la que se apoyará el sistema. Dicha arquitectura (al igual que los estándares escogidos) impondrá numerosos requisitos al desarrollo del sistema pero garantizará que la solución sea abierta e interoperable. Teniendo en cuenta que esta metodología se centra en el dominio sanitario y que las contribuciones presentadas apuntan a la norma ISO12967 (junto con RM-ODP) como los esfuerzos normalizadores más completos, se ha escogido HISA como la norma más adecuada en este contexto (aunque podrían utilizarse otras iniciativas). En el Apartado 2 se lleva a cabo la conformidad de HISA con respecto a RM-ODP para su establecimiento como marco arquitectural del trabajo.

Fase 4. Ciclo iterativo: revisión incremental

En este punto se sustituye el proceso en cascada por uno iterativo en el que cada ciclo desarrolla de forma incremental el sistema. Al final de cada ciclo se obtendrá un prototipo completamente funcional y validado aunque no con todas las capacidades del sistema final que se busca desarrollar. La primera fase de cada ciclo es una revisión incremental en la que se define el alcance del prototipo a desarrollar en ese ciclo y se analizan los objetos desarrollados hasta el momento. Es una fase con un doble propósito: permite planificar el ciclo siguiente especificando exactamente qué partes del sistema van a ser desarrolladas y, además, sirve como fase de evaluación del proceso mejorando la eficiencia del mismo al buscar y corregir errores cometidos en los ciclos anteriores.

Fase 5. Ciclo iterativo: análisis

Una vez definido el prototipo a desarrollar en el ciclo se procede con el análisis del mismo y la definición de sus propiedades esenciales. Se parte de los casos de uso, actores y requisitos definidos en las fases 2 y 3 pero sólo de aquellos implicados en este prototipo (si, por ejemplo, no se consideran en este ciclo las comunicaciones externas del sistema, no habrá que tener en cuenta los protocolos estándares de comunicación). Los casos de uso y procesos son ahora desarrollados en detalle y se profundiza también en la funcionalidad del prototipo. Esta fase corresponde a las vistas independientes de tecnología de RM-ODP o al modelo independiente de plataforma (PIM) de la filosofía MDA. Los resultados obtenidos presentan el prototipo independientemente de la tecnología de implementación lo que se traduce en un

conjunto de posibles soluciones tecnológicas para el mismo análisis. En sucesivas iteraciones de esta fase se ha desarrollado el trabajo presentado en los Apartados 4.2 y 5.

Fase 6. Ciclo iterativo: diseño

En la fase de diseño se concreta el prototipo para una tecnología concreta definiendo las vistas dependientes de tecnología de RM-ODP o el modelo específico de plataforma (PSM) de MDA. Esta fase sirve entonces como planificación de la implementación y debe tener en cuenta todos los requisitos especificados en las fases anteriores pues la tecnología debe satisfacerlos. A esta fase corresponden las contribuciones del Apartado 6.4.

Fase 7. Ciclo iterativo: implementación

La fase de implementación está centrada en el desarrollo del prototipo e incluye, según sea el caso la generación de código, integración con otros elementos ya desarrollados, construcción de módulos software, etc. En esta fase y como parte de la implementación, se realizan pruebas aisladas en entornos controlados de los elementos que han sido creados. En el Apartado 6.4.3 se pueden encontrar los resultados de esta fase.

Fase 8. Ciclo iterativo: pruebas

En esta última fase del ciclo iterativo se integra el prototipo con los elementos, arquitectura, etc., con los que debe cooperar y se valida que responde a la funcionalidad que se definió en la revisión incremental. Además se analiza el resultado implementado y se identifican errores y funcionamientos anómalos. Si el resultado de esta fase es un prototipo del sistema final entonces se vuelve a entrar en un nuevo ciclo a través de una nueva fase de revisión incremental. Si por el contrario el prototipo es el sistema completo, se termina con el proceso en espiral y se pasa a la fase 9.

Fase 9. Aceptación del sistema

En esta última fase, y una vez desarrollado el sistema completo, puede desplegarse en el entorno real y evaluar el impacto que tiene sobre los procesos sanitarios. Será necesario considerar en esta fase la realización de actividades orientadas a la modificación de los procesos sanitarios con la inclusión del nuevo sistema y la aceptación del mismo por los actores sanitarios implicados.

2. Resultado 1 - La arquitectura sanitaria basada en HISA

Una de las principales barreras para la evolución del escenario sanitario es que la integración, interoperatividad y reutilización de los servicios son tareas complejas en un entorno tan heterogéneo y cambiante. Aunque la normalización es clave para la interoperatividad, las capacidades y terminologías gestionadas por los servicios son heterogéneas en su naturaleza a pesar de utilizar ontologías e interfaces normalizadas. Sólo a través de la completa comprensión de toda la arquitectura se puede alcanzar una integración y composición de servicios efectivas que proporcionen capacidades avanzadas al usuario final. Una arquitectura sanitaria podría ser definida como una descripción formal de un sistema que proporciona servicios sanitarios, sociales y del bienestar, organizada de manera que permite razonar sobre las propiedades estructurales del sistema. Se definen componentes del sistema, o bloques constructivos, y se proporciona un plan sobre el que desarrollar productos y otros sistemas. Además proporciona el marco de referencia adecuado para guiar la estrategia de negocio sanitario así como el diseño, desarrollo y despliegue de componentes dentro de un sistema de acuerdo a objetivos y requisitos. El diseño y la formalización de tal arquitectura deben ser realizados usando paradigmas estándar, métodos y lenguajes formales alcanzados por consenso y teniendo en cuenta los esfuerzos de estandarización existentes para facilitar de forma real la reutilización y escalabilidad de los servicios.

El establecimiento a través de la normalización de una arquitectura sanitaria que cubra (o al menos considere) los principales escenarios sanitarios permitirá crear un marco de trabajo en el que todos los actores implicados encontrarán herramientas, guías y patrones para enfrentar los problemas y requisitos a la vez que se mantienen los niveles de interoperatividad y escalabilidad. Tal marco de trabajo debe establecer de forma coherente el conjunto de estándares, métodos y mejores prácticas para el desarrollo, validación y también certificación de servicios y aplicaciones complejos. Debe considerar asimismo las infraestructuras y servicios sanitarios más innovadores y relevantes, reutilizables en diferentes escenarios y casos de uso. Es necesario identificar a su vez solapamientos o vacíos en los estándares y métodos que necesiten ser salvados para establecer una ordenación de guías coherente. Con todas estas consideraciones este marco sería el pilar para el diseño de infraestructuras que permitirán abordar el cambio de paradigma sanitario, abriendo nuevas líneas de negocio para la provisión de la asistencia más personalizada posible. La norma ISO/EN 12967 (HISA) [2] y otras propuestas arquitecturales particularizadas para el dominio sanitario (como los proyectos HSSP [105] o HITSP [146]) parecen un interesante primer paso en el camino. En este trabajo de Tesis Doctoral se ha considerado RM-ODP como el marco de referencia que mejor encaja con los objetivos perseguidos y como su particularización en el dominio sanitario se utiliza la norma HISA que sirve de base de la arquitectura sanitaria de sistemas distribuidos y abiertos. En este apartado se presenta la alineación de la norma HISA con el marco de referencia RM-ODP y se formalizan sus conceptos utilizando el perfil UML4ODP [27]. En los Apartados 4 y 5 se extenderá la norma con aspectos de seguridad y gestión semántica normalizados.

Aunque RM-ODP es una importante base para la producción de HISA ya que es el marco de referencia sobre el que se apoya la norma, sólo los tres puntos de vista independientes de tecnología (es decir, de la Empresa, de la Información y Computacional) han sido utilizados para producir el estándar. Los dos puntos de vista dependientes de tecnología deben ser considerados dentro de un contexto de implementación específico y, como consecuencia, no han sido estandarizados. RM-ODP define conceptos para la especificación formal de cualquier sistema distribuido pero no fuerza a utilizar un lenguaje de modelado concreto para su representación aunque el perfil de UML para ODP está progresivamente extendiéndose como lenguaje normalizado de modelado. Por su parte, HISA ha surgido como una referencia relevante para el desarrollo de arquitecturas sanitarias pero no está adquiriendo el ímpetu y la aceptación que se supusieron a priori. En este trabajo de Tesis Doctoral se ahonda en la hipótesis que establece que si HISA estuviera más alineada con los principios de RM-ODP (no sólo en la separación de puntos de vista) posiblemente su aplicación sería más sencilla y estaría más extendida. De hecho HISA no utiliza los lenguajes de puntos de vista de RM-ODP ni está especificada de manera formal.

Cabe destacar que RM-ODP e HISA no tienen exactamente el mismo dominio de aplicación ya que el primero conduce al desarrollo de sistemas distribuidos mientras que HISA es la especificación de una arquitectura en sí misma. Teniendo esto en cuenta, se ha refinado la formalización de las tres vistas independientes de tecnología de HISA buscando su conformidad con RM-ODP y el estándar UML4ODP.

2.1. Afinidades y divergencias entre HISA y RM-ODP

La primera versión de HISA fue publicada en 1997 y especificaba una arquitectura de servicios para el dominio sanitario cuyos principios arquitecturales estaban formalizados de acuerdo a los criterios especificados por RM-ODP. Por su parte, tras diez años de uso público y realimentación, el modelo de referencia ODP había llegado a ser una propuesta madura de estandarización con amplia diseminación y aceptación en la industria y la investigación [29][177]. HISA adopta el uso de puntos de vista para facilitar la separación de los diferentes aspectos funcionales y de negocio que confluyen en un mismo sistema, heredando así una de las aportaciones más útiles de RM-ODP y de las recomendaciones internacionales [17]. Desde su concepción HISA no pretende ser un conjunto completo y cerrado de especificaciones sino una arquitectura abierta y con capacidad de evolución; por eso formaliza sólo aspectos concretos identificados como comunes y esenciales de todos los sistemas de información sanitarios avanzados. Esta flexibilidad tiene como consecuencia que ciertas áreas no estén cubiertas en detalle como, por ejemplo, algunos aspectos de seguridad [95]. Durante los diez años que HISA ha sido un pre-estándar europeo varios trabajos han señalado puntos débiles de la norma tales como la difusa separación de los puntos de vista de la Empresa y la Información o las dificultades para distinguir entre sus partes normativas e informativas [178][179]. Muchos de estos aspectos han sido mejorados para la versión final del estándar pero, aun teniendo en cuenta los diferentes niveles de aplicación de HISA y RM-ODP, la arquitectura de servicios sanitarios no adopta estrictamente los criterios definidos en el

modelo de referencia ODP. En este apartado se realiza una revisión de las principales convergencias y divergencias entre ambos estándares con la intención de plantear posteriormente contribuciones para la completa alineación de HISA con RM-ODP.

La primera divergencia que un arquitecto encuentra entre ambos estándares es la metodología de especificación de puntos de vista. RM-ODP no prescribe un proceso concreto para realizar dicha tarea, trasladando esta decisión al arquitecto. Este punto ha sido criticado en algunos trabajos [180] e incluso existen propuestas para combinar RM-ODP con conocidas metodologías o construir otras en exclusiva [181][182]. HISA incluye algunos detalles para el proceso de desarrollo estableciendo una metodología en dos partes:

- Uso del paradigma estratégico como guía para la especificación de los puntos de vista de la Empresa, de la Información y Computacional. Este documento tiene que ser conciso, orientado a la gestión e identificar requisitos y objetivos estratégicos del sistema. El punto de vista de la Empresa es especificado a partir del paradigma estratégico, y los puntos de vista de la Información y Computacional son especificados en un proceso en cascada.

- Un proceso iterativo para la especificación de cada punto de vista a través de múltiples niveles de refinamiento.

Desde la perspectiva del arquitecto HISA proporciona una guía para la especificación de la arquitectura mientras que RM-ODP es neutral en cuanto a los procesos de diseño y desarrollo lo que permite que un arquitecto pueda elegir cualquier metodología para especificar los sistemas. A priori es difícil concluir qué metodología es la más apropiada para una arquitectura en el dominio sanitario y sólo a través de la experiencia y el uso de los marcos de trabajo se puede extraer una conclusión fundamentada. De acuerdo con las lecciones aprendidas de otras propuestas se puede optar en primera instancia por un proceso de refinamiento iterativo. En un primer paso se especifican los puntos de vista de forma general y abstracta para, a continuación, en un número indefinido de pasos, refinar y concretar cada punto de vista con mayor detalle y precisión. Este podría ser un proceso de especificación adecuado para un dominio tan complejo y amplio como es el sanitario.

La notación y los lenguajes de formalización y modelado son el siguiente punto en el que estas normas presentan diferencias. RM-ODP fue publicado sin referencia normativa en cuanto a notación y posteriormente aparecieron el lenguaje de empresa (ISO 15414) [23] y UML4ODP (cuatro y diez años después, respectivamente). Esta prolongada espera por una notación normalizada explica, al menos en parte, por qué los requisitos especificados por HISA no siguen un formato estandarizado. De hecho el estándar no prescribe la adopción de ningún tipo de herramienta de modelado o lenguaje aunque se recomienda el uso de UML. La propuesta actual de HISA utilizando diagramas UML y texto plano puede ser útil para arquitecturas pequeñas y simples pero es inviable para cubrir un sistema sanitario complejo, flexible y cambiante en el tiempo. Además la utilización de notaciones y lenguajes

normalizados permite reducir las tareas de gestión a lo largo de todo el ciclo de vida de los sistemas y facilita asimismo el intercambio de especificaciones y la interoperatividad entre organizaciones y arquitectos. HISA no considera las normas ISO 15414 y UML4ODP como referencias normativas sino sólo como bibliografía de interés. En esta línea, una de las contribuciones del presente trabajo es la transferencia de requisitos y especificaciones normativas de HISA a una formalización conforme a RM-ODP y los estándares ISO 15414 y UML4ODP.

Otra consideración de la etapa de formalización es la especificación de correspondencias entre puntos de vista. HISA describe sólo unas pocas relaciones entre los puntos de vista de la Empresa y la Información y siempre en texto plano. Las correspondencias entre los puntos de vista de la Información y Computacional están más claramente establecidas pero HISA no tiene suficientemente en cuenta este aspecto de la formalización de arquitecturas que resultaría en una especificación coherente e integrada con todas las vistas trabajando juntas sin inconsistencias. En contraposición, la especificación de correspondencias entre puntos de vista es uno de los puntos fuertes de RM-ODP como ha sido señalado en muchos trabajos [183][184]. Este marco establece que la completa especificación de un sistema debería incluir todas las correspondencias entre términos así como constructores formales que permitan relacionar los puntos de vista. RM-ODP sólo describe correspondencias genéricas entre los puntos de vista de la Información y Computacional, y entre el Computacional y el de Ingeniería. Para el resto de parejas de puntos de vista las correspondencias dependerán de las especificaciones del sistema concreto. La notación de correspondencias está ampliamente cubierta en el estándar UML4ODP mediante la introducción del concepto “enlace de correspondencia” (*correspondence link*). En la formalización de HISA a través del perfil de UML4ODP que se ha realizado para esta Tesis se establecen de manera formal las correspondencias especificadas por la norma entre los diferentes puntos de vista concluyendo que HISA podría adoptar esta notación formal de manera directa fortaleciendo así sus fundamentos normativos.

Una vez que un sistema está especificado e implementado, su conformidad con el estándar y la correspondiente especificación debe ser testeada. Más allá de la definición de correspondencias, el proceso de evaluación de la conformidad es otro punto que HISA y RM-ODP abordan de manera diferente. La primera establece que una adecuada especificación de un sistema sanitario será conforme con la metodología y contendrá los documentos obligatorios (es decir, el paradigma estratégico y los puntos de vista de la Empresa, la Información y Computacional). La especificación de los puntos de vista de Ingeniería y Tecnología es opcional. Además un sistema conforme a HISA tendrá que implementar todos los objetos de información y proporcionar todos los servicios computacionales definidos en el estándar. Por su parte RM-ODP identifica puntos de conformidad dentro de las especificaciones de los puntos de vista en los cuales se pueden realizar evaluaciones de la conformidad. También define clases de puntos de conformidad y especifica la naturaleza de las sentencias de conformidad. Una razón por la cual HISA no adopta estos puntos es porque están principalmente definidos en el punto de vista de Ingeniería y la norma sanitaria sólo establece como obligatorios los tres puntos de vista independientes

de tecnología. Así, en la actual versión del estándar, sólo puede evaluarse la conformidad de aquellos aspectos que se especifiquen en alguno de estos tres puntos de vista y no de aquellos que aparezcan en los puntos de vista dependientes de tecnología.

Finalmente, el último punto que merece ser estudiado es la estandarización de los servicios de infraestructura. De acuerdo con la identificación de servicios básicos RM-ODP define funciones fundamentales para la construcción de cualquier sistema ODP. Estas funciones son servicios arquitecturales básicos que serán incluidos en el diseño de la implementación y están agrupados en las siguientes áreas: gestión, coordinación, repositorio y seguridad. Actualmente sólo dos funciones están estandarizadas, la función de intermediación (*trading function)* [24] y la de repositorio de tipos (*type repository function)* [25] pero todas son consideradas y expresadas por interfaces y estereotipadas en el perfil UML del punto de vista de Ingeniería definido en el estándar UML4ODP. Estas funciones proporcionan funcionalidades de propósito general para la construcción de sistemas ODP y aunque no están todas desarrolladas en el marco de RM-ODP, se considera que su identificación es un punto inicial útil que HISA debería adoptar. Sin embargo, la norma EN/ISO 12967 no las menciona explícitamente en ningún momento aunque podría considerarse que se adoptan implícitamente ya que HISA determina que se seguirán los fundamentos establecidos por RM-ODP.

2.2. La formalización de la arquitectura sanitaria basada en HISA a través del perfil de la ISO 19793 (UML4ODP)

En este apartado se da el primer paso en la especificación de la arquitectura sanitaria de referencia que soportará el escenario POVO. Los aspectos fundamentales de esta arquitectura van a venir heredados de la norma HISA y a estos cimientos se le añadirán los aspectos de seguridad y gestión de semántica. Como el estándar no sigue la descripción formal de RM-ODP (UML4ODP e ISO 15414), en este apartado se especifican las tres vistas independientes de tecnología descritas por HISA mediante los lenguajes y notaciones de dichas normas. Antes de empezar a analizar los diferentes puntos de vista, es necesario especificar los diagramas de alto nivel de la arquitectura. Como se desprende de la Figura 5.4, y siguiendo la recomendación X.906, el *HISA System* está especificado en un paquete estereotipado *ODP_SystemSpec* que contiene un paquete por cada vista y uno más por cada especificación de correspondencia entre vistas. Formalizando exclusivamente la norma HISA nos lleva a tener las vistas de la Empresa, la Información y Computacional así como las correspondencias entre ellas.

2.2.1. Punto de vista de la Empresa

Las partes normativas del estándar HISA para este punto de vista son: un esquema general con los principales objetos y sus relaciones, un conjunto de requisitos de información generales, tres grupos de requisitos por flujos de trabajo (sujeto asistido, información clínica y gestión de actividad) y especificaciones de requisitos para clústeres de actividades de usuarios (actividades de gestión de recursos, de gestión de usuarios y autorizaciones, y de gestión de clasificaciones, codificaciones y

diccionarios). Todos estos puntos normativos del estándar han sido especificados en la arquitectura mediante diagramas del perfil UML para RM-ODP.

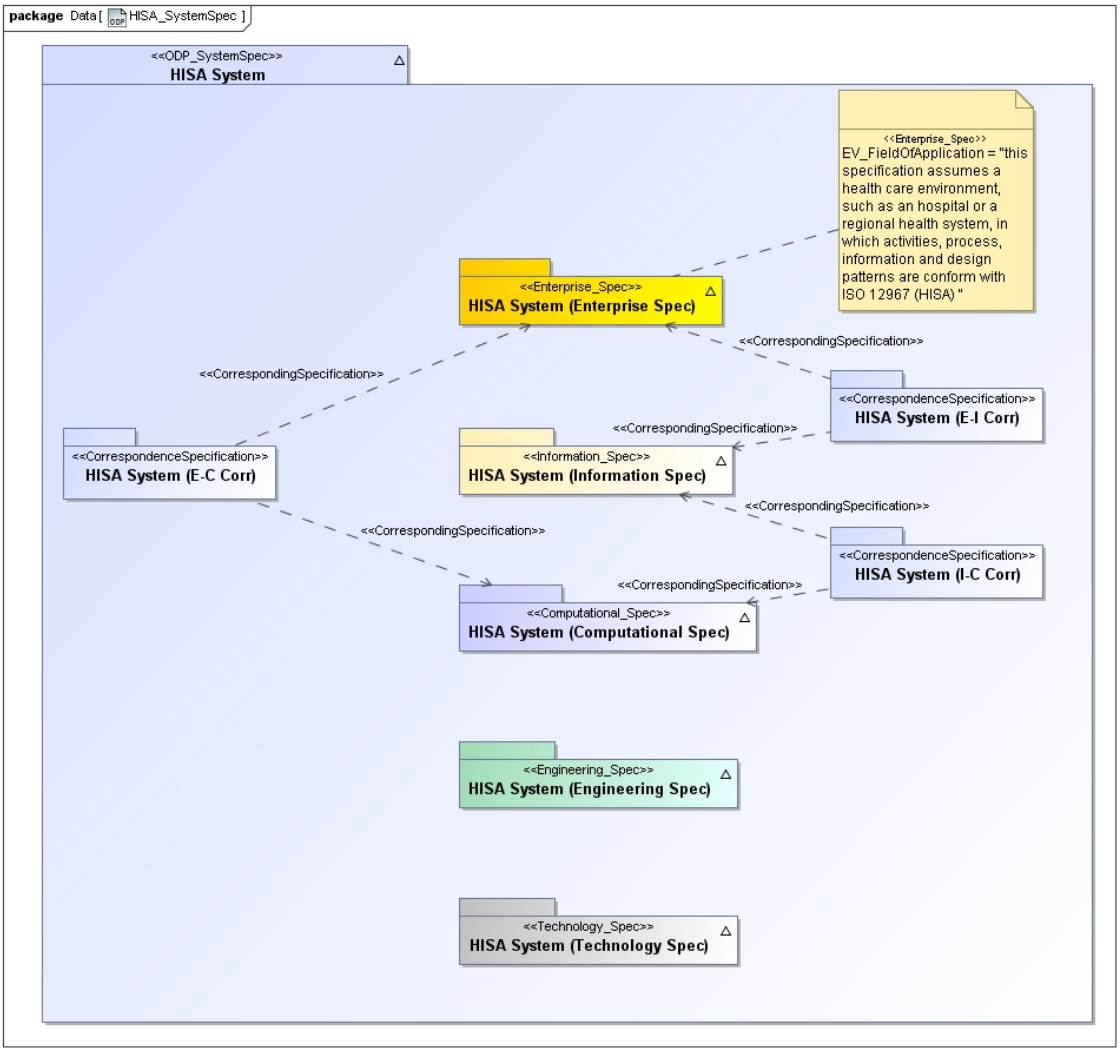

Figura 5.4. Paquetes de especificación de alto nivel de la arquitectura y relaciones

En primer lugar se ha especificado una comunidad llamada *Healthcare Information System* (HIS) cuyo objetivo, extraído del paradigma estratégico de HISA, es soportar las actividades de asistencia sanitaria, organizacionales y de gestión dentro de una organización sanitaria. Esta comunidad engloba todos los elementos que forman parte de la arquitectura definida por HISA. La propuesta adoptada es considerar esta comunidad como una agregación de comunidades (Figura 5.5) consiguiendo así una separación de capacidades funcionales en distintas comunidades mientras que se mantiene la noción de comunidad global o sistema integral. Estas comunidades son sólo refinamientos de las diferentes áreas de la comunidad HIS.

En lo que respecta a la norma HISA se ha definido una comunidad por cada flujo de trabajo y clúster de actividades. En este apartado se aborda la formalización de la comunidad HIS como un todo definiendo roles, objetos y demás aspectos de empresa sin diferenciar a cuál de estas comunidades pertenece ya

que el desarrollo de cada comunidad por separado conllevaría una pérdida de sentido integral con respecto a lo que normaliza la ISO/EN 12967. Las extensiones que se realizarán a la norma en los sucesivos apartados se materializarán en dos comunidades (una para seguridad y otra para las capacidades de gestión de semántica) dentro de la agregación de la comunidad HIS. Estas extensiones sí se formalizan de forma separada por el interés y complejidad que presentan por sí mismas. Pese a ello se detallará cómo dichas comunidades se enlazan con los aspectos definidos en la norma y recogidos en alguna de las comunidades de la Figura 5.5.

En las actividades de los usuarios se definen tres flujos de trabajo fundamentales:

1. Flujo de trabajo del Sujeto Asistido: relacionado con las actividades del usuario centradas en la gestión de información personal y estadística de los sujetos asistidos y la gestión de sus encuentros con la organización.

2. Flujo de trabajo de Gestión de Actividad: agrupa las actividades de los usuarios relacionadas con la gestión de diferentes tipos de actividades ejecutadas en la organización durante todo su ciclo de vida incluyendo la solicitud inicial, reserva, planificación, ejecución e informe de las mismas.

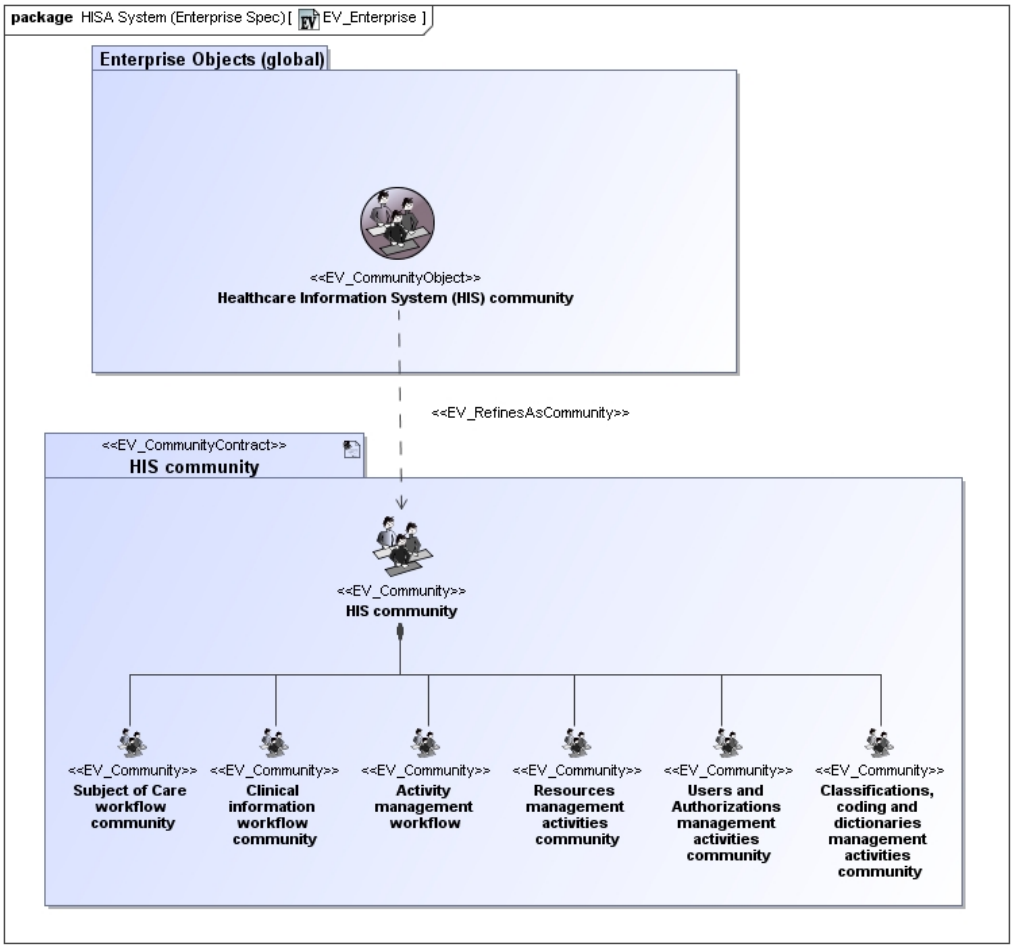

Figura 5.5. Formalización de la comunidad HIS como agregación de comunidades

3. Flujo de trabajo de Información Clínica: actividades de usuarios relacionadas con la gestión de datos clínicos.

Otros elementos soportarán las siguientes agrupaciones de actividades de usuarios:

- Gestión de la información relacionada con la descripción de la estructura de la organización, de los usuarios y de los criterios de autorización de acuerdo a los cuales se les permite a los usuarios acceder a los datos y ejecutar funcionalidades.

- Gestión de recursos disponibles en la organización, y las reglas y criterios conforme a los cuales están disponibles y pueden ser utilizados.

- Gestión de clasificaciones, criterios de codificación y diccionarios adoptados para clasificar la información gestionada.

En cuanto a los procesos, que de acuerdo con RM-ODP modelan secuencias de acciones llevadas a cabo por objetos de empresa, HISA no describe ninguno de forma obligatoria sino sólo a título informativo por tanto no se han incluido en la formalización del punto de vista de la Empresa. El segundo punto consiste en la identificación de los roles y actores y las reglas de asignación entre ellos. Un sistema y un rol han sido especificados para cada comunidad (esto es, para cada flujo de trabajo y clúster de actividades) de acuerdo con la información que proporciona HISA, aunque un refinamiento o extensión de estas comunidades llevaría a la identificación de un mayor número de roles y actores. Paralelamente, otros objetos de empresa como *PatientPlan* o *Resource* han sido identificados y se ha añadido además el rol genérico *User* extraído del paradigma estratégico. Como no se definen procesos en la norma, tampoco existen interacciones entre roles que modelar. El siguiente paso es la identificación de los objetos de empresa que se pueden agrupar en individuos (*Healthcare provider*, *Individual User* y *Subject of care),* objetos (*Resource*, *Clinical Information*, *Authorization profile*, *Clinical Guidelines & Protocols*, *PatientPlan* y *Clinical Plan)* y sistemas (uno por cada flujo de trabajo y clúster definido en la norma). Al igual que ocurre con los roles, como no existen procesos en HISA no podemos especificar los estados de los objetos de empresa.

Lo que queda por modelar son las políticas del punto de vista de la Empresa que especifican restricciones en la estructura o comportamiento de la comunidad. La norma HISA establece, por un lado, los requisitos de información generales para las actividades de todos los usuarios que tienen en cuenta los atributos mínimos que deben incorporar todos los objetos de información para cubrir requisitos de descripción, extensibilidad, control de versiones, auditorías y control del ciclo de vida. Por otro, para cada flujo de trabajo se cuenta con los siguientes requisitos:

1. Flujo de trabajo del sujeto asistido:

    a. Debe existir un identificador único para cada paciente en todos los servicios de información.

b. Considerar el término *paciente* contenido en *sujeto asistido* el cual tiene un ciclo de vida más amplio.

2. Flujo de trabajo de información clínica:

   a. La información clínica puede tomar la forma de datos elementales o de agregaciones de múltiples datos elementales.

   b. Cada pieza de información debe estar presente sólo una vez en el sistema para evitar la necesidad de múltiples entradas y el riesgo de inconsistencias.

   c. Se deben establecer relaciones entre los datos y las actividades para mejorar la calidad y fiabilidad de los tratamientos y posibilitar la monitorización de costos y la calidad de la organización.

   d. Es necesario que existan funcionalidades autónomas que gestionen el flujo de información clínica sin depender de la ejecución de actividades.

3. Flujo de trabajo de gestión de actividades:

   a. Debe ser posible relacionar cada actividad con las necesidades de un paciente individual.

4. Actividades de gestión de recursos:

   a. No hay requisitos más allá de la existencia obligatoria de mecanismos comunes de definición y recuperación de información sobre los recursos disponibles y reglas o criterios de gestión y uso de los mismos.

5. Actividades de gestión de usuarios y autorizaciones:

   a. Necesidad de diversificar roles y responsabilidades.

   b. Necesidad de asegurar los datos gestionados.

   c. Necesidad de controlar y monitorizar la autorización de usuarios individuales para ejecutar ciertas actividades del sistema.

   d. Posibilidad de definir reglas y criterios de acceso al sistema y ejecución de actividades para cada usuario individual, de acuerdo a su rol y responsabilidad.

   e. El software de intermediación proporcionará un repositorio con definiciones de reglas según las cuales los diferentes usuarios pueden ejecutar funciones del sistema, así como mecanismos estándar de comprobación de acceso.

f. Para cada componente del sistema (aplicaciones y servicios, los recursos se contemplan en el punto 4) se debe definir un conjunto de perfiles de autorización.

6. Actividades de gestión de clasificaciones, codificaciones y diccionarios:

   a. Para cada flujo de trabajo y agrupación de actividades el software de intermediación debe permitir la definición y gestión de diccionarios, clasificaciones y reglas adaptados para ese proceso.

   b. Debe proporcionarse un soporte para la gestión de dependencias y relaciones entre conceptos.

   c. Todos los servicios del sistema deben gestionar su propia información, así como diccionarios y clasificaciones de términos en su ámbito de actividades.

Cuatro políticas de empresa han sido definidas de acuerdo a los requisitos directamente extraídos del punto de vista de la Empresa de HISA. Éstas son:

- *Identification of subjects of care*: describe el requisito (1.a) "debe existir un identificador único para cada paciente en todos los servicios de información".

- *Non-duplicity of clinical information*: requisito (2.b) "cada pieza de información debe estar presente sólo una vez en el sistema, para evitar la necesidad de múltiples entradas y el riesgo de inconsistencias".

- *Authorization profile for users*: requisito (5.d), "posibilidad de definir reglas y criterios de acceso al sistema y ejecución de actividades para cada usuario individual, de acuerdo a su rol y responsabilidad".

- *Authorization profile for resources*: requisito (4.a), "existencia obligatoria de mecanismos comunes de definición y recuperación de información sobre los recursos disponibles y reglas o criterios de gestión y uso de los mismos".

La traslación de requisitos de HISA descritos en texto plano a políticas del punto de vista de la Empresa ha presentado algunas dificultades principalmente al no establecer claramente qué requisitos son estrictamente obligatorios y cuáles son ejemplos utilizados para ilustrar el texto de la norma. Al margen de las cuatro políticas identificadas, otros requisitos comentados podrían extraerse de HISA pero su descripción formal implicaría introducir objetos de empresa y procesos no considerados explícitamente en el estándar. Aunque estos requisitos no pueden por tanto ser descritos como políticas, tienen que ser considerados en los puntos de vista dependientes de tecnología los cuales quedan fuera del alcance de la norma.

2.2.2. Punto de vista de la Información

El punto de vista de la Información de HISA especifica varios diagramas de clase UML que identifican objetos de información y atributos requeridos para cada uno de ellos. En esta especificación inicial de la arquitectura el punto de vista de la Información consiste en un esquema invariante por cada diagrama UML de HISA obteniendo así un esquema de alto nivel de los objetos de información (Figura 5.6), dos de metamodelado (estos son, clase HISA genérica y objetos de clasificación) y seis que corresponden a flujos de trabajo y clústeres de actividades. De acuerdo con RM-ODP, un punto de vista de la Información puede contener también esquemas estáticos y dinámicos. Se ha considerado que este es un punto clave que merece ser desarrollado en detalle para poder entender los diferentes estados por los que pueden pasar los objetos de información. Algunos objetos (por ejemplo, *Agent* o *Resource*) pueden no tener un único y definido conjunto de estados y podría variar de una implementación a otra. En estos casos puede entenderse que no se especifique un grupo de estados concreto pero no en otros como el objeto de información *Activity*. Este elemento tiene su correspondencia en el objeto de empresa homónimo y ahí sí está definido un conjunto de estados, concretamente *foreseen*, *requested*, *accepted*, *terminated* y otros.

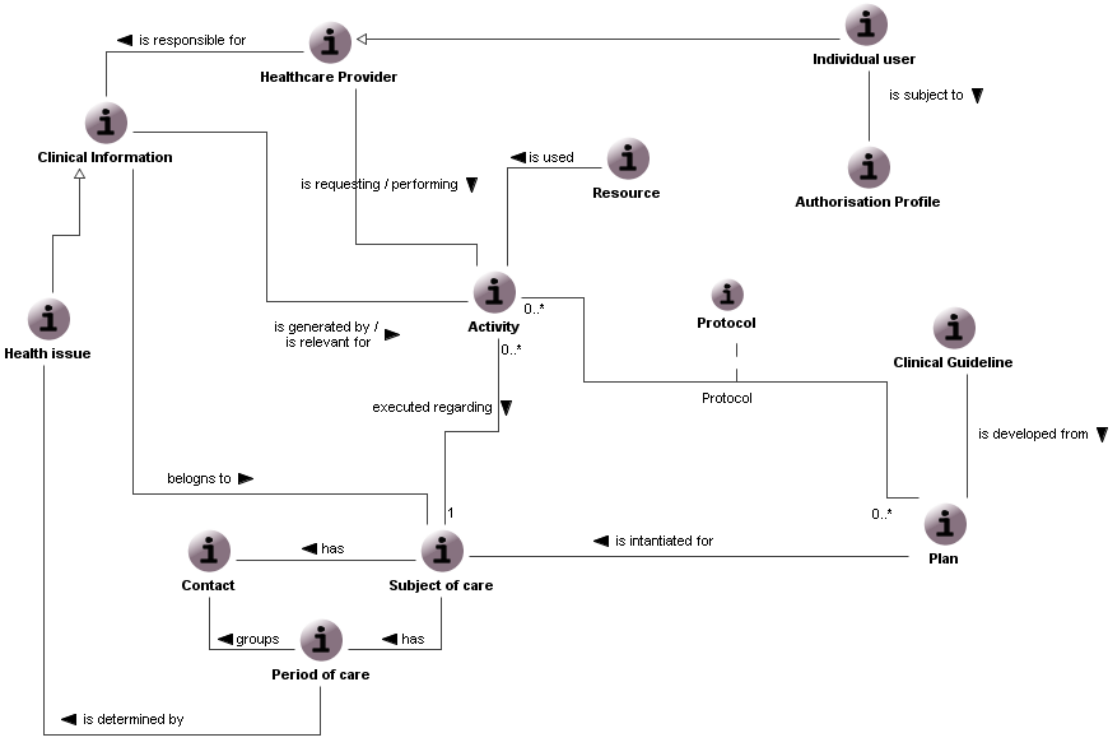

Figura 5.6. Modelo de alto nivel de los objetos de información

2.2.3. Punto de vista Computacional

La tercera parte del estándar ISO/EN 12967 describe tres tipos de objetos computacionales: básicos, de propósito general y complejos. Cada uno tiene un conjunto de métodos genéricos asociados así como una descripción textual y un alcance. La especificación de estos requisitos en nuestra arquitectura ha

dado como resultado un diagrama incluyendo los tres objetos computacionales, sus métodos e interfaces (Figura 5.7). Para cada clase de información especificada debe dotarse al software de intermediación con un objeto computacional básico que implemente los métodos estandarizados por la norma HISA. Estos métodos permitirán el acceso y el manejo de cada concepto (es decir objetos y propiedades) de la clase. Por otro lado, la interfaz de propósito general y las interfaces complejas deben ser aportadas por el software de intermediación al menos en un objeto computacional para cada una de ellas dentro del sistema. Este punto de vista tendrá que ser forzosamente extendido cuando se cubran otras áreas como las que se tratan a continuación y será especialmente valioso reutilizar interfaces de objetos normalizadas provenientes de iniciativas o recomendaciones extendidas.

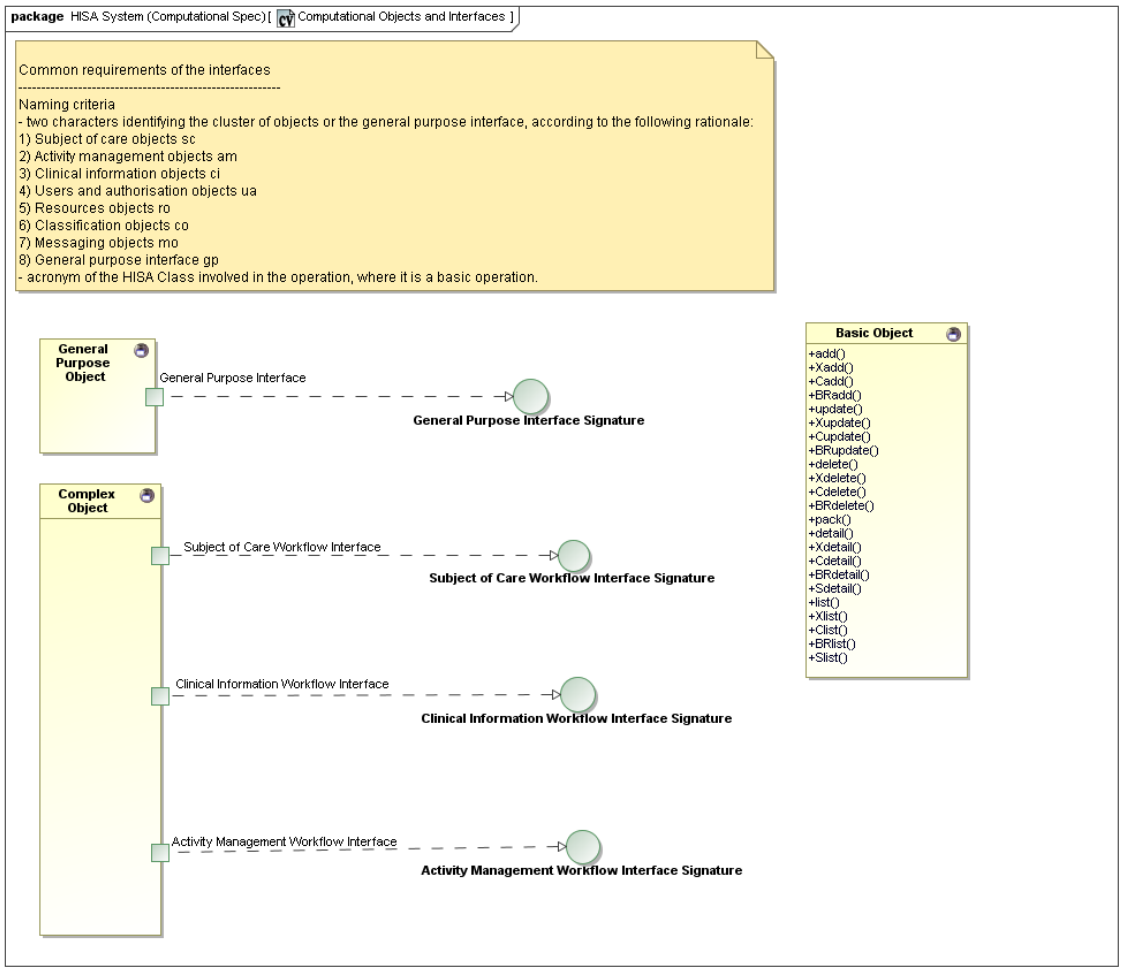

Figura 5.7. Objetos computacionales e interfaces de la norma HISA

2.2.4. Correspondencias entre vistas

La aplicación de la notación del perfil UML para RM-ODP a los fundamentos de HISA ha sido sencilla a causa de la simplicidad de los puntos de vista de la norma. HISA no describe explícitamente correspondencias entre los puntos de vista de la Empresa y la Información pero algunas relaciones entre objetos pueden ser extraídas. Este mapeo entre objetos de empresa e información es inmediato ya que

ambos puntos de vista están construidos sobre el mismo modelo de alto nivel y cuya fuente es el paradigma estratégico. Un paquete estereotipado *CorrespondenceSpecification* contiene nueve *CorrespondenceLink* entre objetos de empresa y de información de la norma. Por otro lado, HISA establece explícitamente correspondencias entre los puntos de vista de la Información y Computacional. En particular, cada objeto computacional básico se corresponde con un objeto de información y los objetos computacionales complejos con flujos de trabajo y clústeres de actividades.

3. Resultado 2 - Armonización de iniciativas de seguridad

La arquitectura que sirve de base para el despliegue del escenario POVO necesita cubrir los aspectos de seguridad y gestión de semántica no abordados en la norma HISA por lo que es requisito indispensable extender ésta de manera formal. En lo que respecta a la seguridad (aunque análogamente se debe hacer para la gestión de semántica) es necesario contar con un marco armonizado de iniciativas de manera que la arquitectura diseñada mantenga su apertura e interoperatividad con otros esfuerzos. En otras palabras, no se trata de desarrollar una arquitectura que rompa con las iniciativas actuales sino de reutilizar todos aquellos esfuerzos normalizadores (el estándar HISA como base es una muestra de ello) que permita construir una solución integradora de los mismos. En este apartado se establece una armonización de marcos de seguridad y control de acceso y se extiende la especificación de la arquitectura del apartado anterior con estos aspectos normalizados.

La principal barrera que existe para la formalización de aspectos de seguridad (en particular para la gestión de privilegios y el control de acceso) en infraestructuras de sistemas distribuidos y heterogéneos es la ausencia de un modelo unificado de autorización que permita integrar las diferentes soluciones desarrolladas y desplegadas en la actualidad. Aunque existen terminologías normalizadoras que han establecido un vocabulario común para los modelos de control de acceso [117][119], en general cada iniciativa nombra de manera específica a sus elementos. Así no hay consensuado un vocabulario estándar para discutir los elementos de modelado del control de acceso a través de plataformas basadas en componentes distribuidos y heterogéneos. Por ejemplo, el RAD del OMG [156] tiene un objeto *AccessDecision* (ADO) que proporciona las *decisiones de acceso* basadas en los *atributos de seguridad de una entidad*, un *recurso con nombre* y una *operación* en el recurso. El modelo de control de acceso de OASIS SAML/XACML [134] define una *AuthorizationAuthority* que proporciona las *decisiones de autorización* basadas en *atributos de un sujeto* y una *acción*. Afortunadamente estas y otras iniciativas contienen elementos arquitecturalmente consistentes lo que significa que siguen los mismos modelos lógicos de autorización y se diferencian en la forma de separar la funcionalidad de autorización en elementos (y su nombrado). Por ejemplo, un objeto *AccessDecision* del OMG y una *AuthorizationAuthority* de OASIS representan el mismo elemento arquitectural en arquitecturas de control de acceso. Como no comparten un modelo de referencia común es difícil determinar sin un análisis en profundidad si proporcionan semántica equivalente o no.

La intención de este apartado es extrapolar esa arquitectura inherente en cada iniciativa a un modelo común normalizado el cual será formalizado conforme a RM-ODP y UML4ODP e integrado en la arquitectura sanitaria diseñada. En concreto se especificarán las tres vistas superiores de la infraestructura de control de acceso en el Apartado 4 y las dependientes de tecnología se abordarán en el Apartado 6 para la validación de la arquitectura con tecnologías concretas y el despliegue del escenario POVO.

3.1. Marco armonizado de seguridad

Pese a que la seguridad es un campo de conocimiento muy complejo y versátil los fundamentos básicos para cualquier sistema de información son generales y compartidos por los distintos marcos de referencia. El ámbito de seguridad se subdivide en disciplinas cada una centrada en un conjunto de requisitos a satisfacer y para los cuales se identifican servicios y mecanismos. Cabe destacar que estas disciplinas no están aisladas; de hecho necesitan colaborar entre sí para establecer el dominio de seguridad completo. A partir de los marcos de referencia descritos anteriormente podemos establecer el conjunto básico de servicios y mecanismos de seguridad identificados:

1. Servicios de **identificación** y **autenticación**
    - mecanismos de cifrado, firma digital, autenticación y claves
2. Servicios de **autorización** y **control de acceso**
    - mecanismos y listas de control de acceso
3. Servicios de **confidencialidad**
    - mecanismos de cifrado, confidencialidad y claves
4. Servicios de **integridad**
    - mecanismos de cifrado, integridad de los datos y claves
5. Servicios de **responsabilidad** y **no repudio**
    - mecanismos de firma digital, integridad de los datos, certificación de validez, responsabilidad, no repudio y registros de auditoría
6. Servicios de **disponibilidad**
    - mecanismos de control de acceso y administración

El glosario de términos que se presenta a continuación está basado en los marcos de seguridad del Capítulo 3 y se utilizará en lo que resta de trabajo. Este glosario no pretende ser un repositorio completo

de todos los conceptos que existen en seguridad sino una ilustración de aquellos términos básicos y generales (comunes a todas las iniciativas y normas) de esta disciplina. Tras la definición de cada concepto se indica la fuente (o fuentes) de la que se ha obtenido y en la Figura 5.8 se ilustra el mapa conceptual de seguridad basado en los principales términos del glosario.

- **Administrador de seguridad** (*security administrator*): persona responsable de la definición o aplicación de una o más partes de una política de seguridad [*X.810*].

- **Auditoría** (*audit*): revisión independiente y examen de los registros y actividades del sistema para verificar la idoneidad de los controles del sistema, asegurar que se cumplen las políticas de seguridad y los procedimientos operativos establecidos, detectar las infracciones de seguridad y recomendar modificaciones apropiadas de los controles, de las políticas y de los procedimientos [*X.800*] [*ISO-2382*].

- **Autenticación** (*authentication*): proceso de confirmación de que una entidad es quien dice ser [*X.800*].

- **Autoridad de certificación** (*certification authority*): autoridad que es confiable (en el contexto de una política de seguridad) para crear certificados de seguridad que contienen una o más clases de datos pertinentes a la seguridad [*X.810*].

- **Autoridad de dominio de seguridad** (*security domain authority*): autoridad de seguridad responsable de la aplicación de una política de seguridad para un dominio de seguridad [*X.810*].

- **Autoridad de seguridad** (*security authority*): entidad responsable de la definición, aplicación o cumplimiento de la política de seguridad [*X.810*].

- **Autorización** (*authorization*): atribución de derechos, que incluye la concesión de acceso basada en derechos de acceso [*X.800*].

- **Certificado de lista de revocación** (*revocation list certificate*): certificado de seguridad que contiene una lista de certificados de seguridad que han sido revocados [*X.810*].

- **Certificado de revocación** (*revocation certificate*): certificado de seguridad expedido por una autoridad de seguridad para indicar que un determinado certificado de seguridad ha sido revocado [*X.810*].

- **Certificado de seguridad** (*security certificate*): conjunto de datos pertinentes a la seguridad expedida por una autoridad de seguridad o tercera parte confiable, junto con información de seguridad que se utiliza para proporcionar servicios de integridad y autenticación [*X.810*].

- **Cifrado** (*encipherment*): transformación criptográfica de datos para producir un criptograma o texto cifrado [*X.800*].

- **Clave** (*key*): parámetro de entrada utilizado para variar la función de transformación utilizada por los algoritmos de cifrado [*IETF*].

- **Confianza** (*trust*): se dice que la entidad X confía en la entidad Y para un conjunto de actividades solamente si la entidad X puede confiar en que la entidad Y se comporta de una manera particular con respecto a dichas actividades [*X.810*].

- **Confidencialidad** (*confidentiality*): propiedad de una información que no está disponible ni es divulgada a persona, entidades o procesos no autorizados [*X.800*].

- **Control de acceso** (*access control*): prevención del uso no autorizado de un recurso, incluida la prevención del uso de un recurso de una manera no autorizada [*X.800*].

- **Credenciales** (*credentials*): (en autenticación) un objeto de datos que asocia un identificador con una unidad de información de autenticación y que puede ser utilizado para verificar la identidad de una entidad que intenta acceder a un sistema; (en control de acceso) un objeto de datos que relaciona un identificador con una o más autorizaciones de acceso y que puede ser utilizado para verificar esas autorizaciones para la entidad que intenta el acceso [*IETF*].

- **Delegación** (*delegation*): el acto por el cual un usuario o entidad autoriza a otro a utilizar su identidad o privilegios quizás con restricciones [*HL7*] [*OMG*].

- **Derecho de acceso** (a*ccess right*): permiso concedido a una entidad para acceder a un objeto particular y para realizar un tipo específico de operación [*ISO-2382*].

- **Disponibilidad** (*availability*): la propiedad de un sistema o recursos de un sistema de ser accesible o utilizable bajo petición por una entidad autorizada de acuerdo a las especificaciones de funcionamiento del sistema [*IETF*].

- **Dominio de seguridad** (*security domain*): un conjunto de elementos, una política de seguridad, una autoridad de seguridad y un conjunto de actividades pertinentes a la seguridad, donde el conjunto de elementos está sujeto a la política de seguridad, para las actividades especificadas y la política de seguridad es administrada por la autoridad de seguridad para el dominio de seguridad [*X.810*].

- **Firma digital** (*digital signature*): datos añadidos a una unidad de datos, o transformación criptográfica de una unidad de datos, que permite al recibiente de la unidad de datos probar la fuente y la integridad de la unidad de datos y proteger contra la falsificación [*X.800*].

- **Granularidad** (*granularity*): en seguridad, nivel de protección. La protección de archivos se considera granularidad gruesa mientras que la protección de campos de los archivos es granularidad fina [*HL7*].

- **Imputabilidad** (*accountability*): propiedad que garantiza que las acciones de una entidad puedan ser rastreadas de manera inequívoca para imputarlas a esa entidad [*X.800*].

- **Información de autenticación** (*authentication information*): información utilizada para establecer la validez de una identidad alegada [*X.800*].

- **Información de seguridad** (*security information*): información necesaria para prestar los servicios de seguridad [*X.810*].

- **Integridad de los datos** (*data integrity*): propiedad que garantiza que los datos no han sido alterados o destruidos de una manera no autorizada [*X.800*].

- **Integridad de los sistemas** (*system integrity*): la calidad de un sistema de procesamiento de datos acometiendo su propósito operacional mientras evita que usuarios no autorizados realicen modificaciones o usen recursos y que los usuarios autorizados realicen modificaciones o usen los recursos de manera inapropiada [*ISO-2382*].

- **Lista de control de acceso** (*access control list*): lista de entidades, con sus derechos de acceso, que están autorizadas a tener acceso a un recurso [*X.800*].

- **Notarización** (*notarization*): registro de datos por parte de un tercero de confianza que permite la ulterior seguridad de la exactitud de sus características, tales como contenido, origen, fecha y entrega [*X.800*].

- **Periodo de acceso** (a*ccess period*): un periodo de tiempo durante el cual son válidos unos derechos de acceso específicos [*ISO-2382*].

- **Política de seguridad** (*security policy*): conjunto de criterios (plan o curso de acción) para la prestación de servicios de seguridad [*X.800*] [*ISO-2382*].

- **Privacidad** (*privacy*): derecho de las personas a controlar o influir sobre la información relacionada con ellos que puede recogerse o almacenarse y las personas a las cuales o por las cuales esta información puede ser revelada [*X.800*].

- **Privilegio mínimo** (*minimum privilege*): restricción de los derechos de acceso de una entidad a únicamente los derechos que son necesarios para la ejecución de las tareas autorizadas [*ISO-2382*].

- **Registro de auditoría** (*audit trail*): datos recogidos que pueden usarse para efectuar una auditoría de seguridad [*X.800*].

- **Reglas de política de seguridad** (*security policy rules*): una representación de la política de seguridad para un dominio de seguridad dentro de un sistema real [*X.810*].

- **Repudio** (*repudiation*): negación de una de las entidades implicadas en una comunicación de haber participado en toda la comunicación o en parte de ella [*X.800*].

- **Sensibilidad** (*sensitivity*): característica de un recurso relativa a su valor o importancia y eventualmente a su vulnerabilidad [*X.800*].

- **Separación de deberes** (*separation of duties*): división de la responsabilidad con la información sensible de manera que un acto individual solamente pueda comprometer la seguridad de una porción limitada de un sistema de procesamiento de datos [*ISO-2382*].

- **Servicio de seguridad** (*security service*): servicio proporcionado por una capa de sistemas abiertos comunicantes, que garantiza la seguridad adecuada de los sistemas y de la transferencia de datos [*X.800*].

3.2. Marco armonizado de control de acceso

Una vez establecido el marco de referencia de seguridad, nos centramos ahora en la disciplina de autorización o control de acceso. A continuación se extrapolan de las iniciativas de control de acceso aquellos puntos comunes que nos permiten establecer un marco de referencia compatible con todos esos esfuerzos. A través de la definición de los conceptos principales obtendremos los fundamentos del control de acceso normalizado. Según el glosario y de acuerdo con la recomendación X.800 [117] tenemos que el concepto *autorización* se define como la atribución de derechos que incluye la concesión de acceso basada en derechos de acceso. A su vez, un *derecho de acceso* es el permiso para una entidad para acceder a un objeto particular y para realizar un tipo específico de operación. La atribución de los derechos de acceso la lleva a cabo una *autoridad de seguridad*. Otros conceptos útiles son: *credenciales de control de acceso*, definidas como objetos de datos que relacionan una entidad con uno o más derechos de acceso y que pueden ser utilizados para verificar esos derechos para la entidad que intenta el acceso; y *delegación*, acto por el cual un usuario o entidad autoriza a otro a utilizar sus derechos de acceso, quizás con restricciones. Entonces el paradigma normalizado de control de acceso se fundamenta en los siguientes puntos:

- se considera un conjunto de entidades que tratarán de acceder y realizar operaciones en un conjunto de objetos o recursos;

- una autoridad de seguridad confiable en el dominio de seguridad será responsable de otorgar derechos a las entidades para acceder a los recursos;

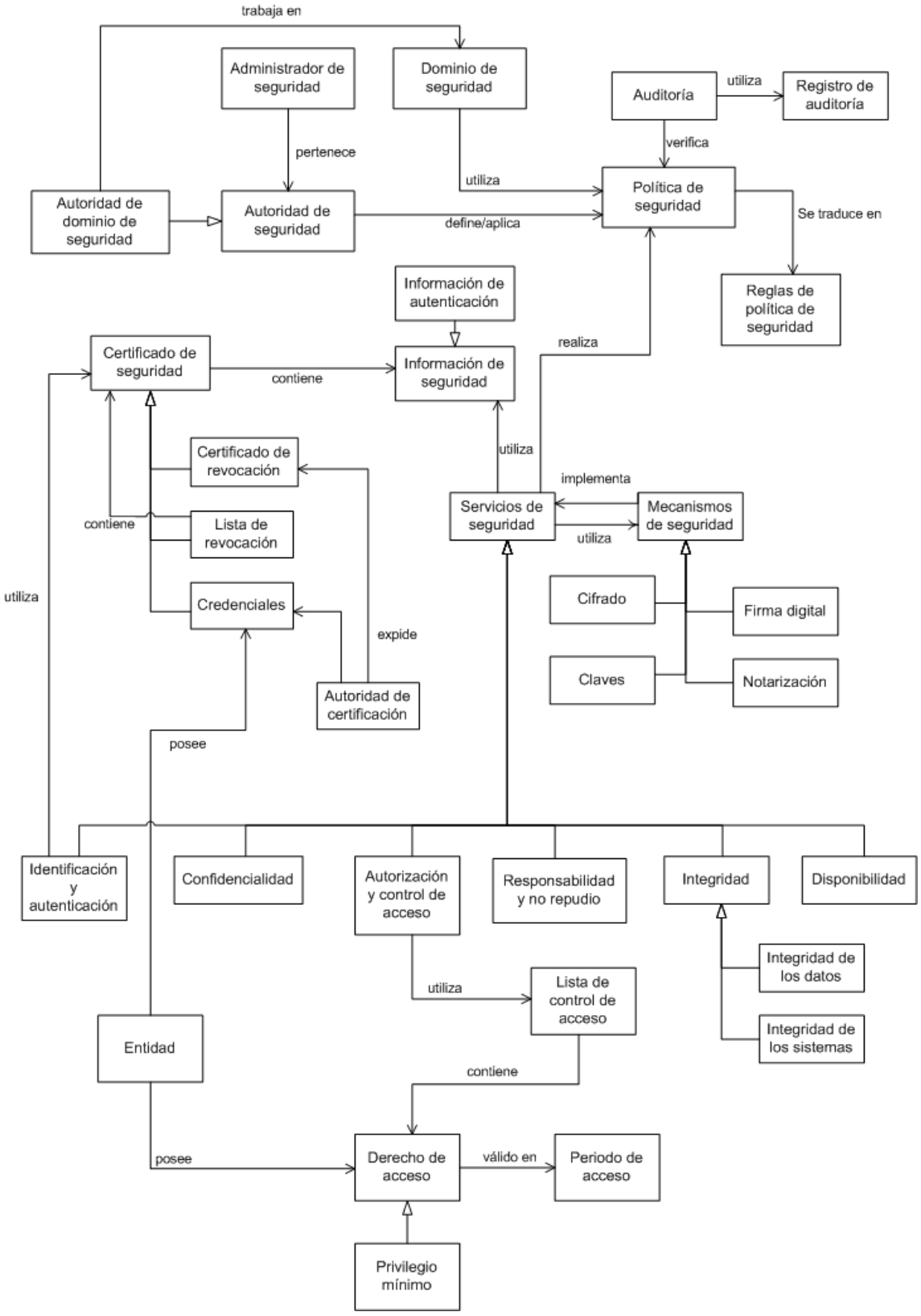

Figura 5.8. Mapa conceptual del dominio de seguridad para el presente trabajo de Tesis Doctoral

- un derecho de acceso identifica a un usuario, un recurso y las acciones que tiene permitido realizar; y

- las entidades podrán delegar sus derechos de acceso (o parte de ellos) a otras entidades.

La Figura 5.9 muestra el esquema básico de control de acceso sustentado en los puntos anteriores. De acuerdo con este esquema para cada par entidad/recurso es necesario establecer y asignar unos derechos de acceso, lo cual hace inviable su utilización práctica en sistemas que cuentan con un número elevado (y/o dinámico) de entidades y recursos. Ante esta falta de escalabilidad del modelo se recurre a flexibilizar la asignación de derechos a través de información de seguridad. Así añadimos las siguientes consideraciones a los fundamentos anteriores:

- cada entidad contará con unos credenciales de control de acceso particulares (objetos de datos de seguridad) asignados por una autoridad de certificación;

- los recursos podrán también ser caracterizados mediante datos de seguridad, asignados por una autoridad confiable de seguridad; y

- los derechos de acceso, en lugar de relacionar entidades con recursos, harán corresponder credenciales de entidades con datos de seguridad de recursos.

Con estas consideraciones se potencia la escalabilidad de las soluciones ya que aunque el número de entidades usuarias y recursos aumente, el conjunto de datos de seguridad tanto para unas como para otros es menos dinámico, permitiendo así la asignación de los mismos datos a muchas entidades o recursos simultáneamente. En la Figura 5.10 se muestra el esquema de control de acceso incluyendo todos los aspectos descritos.

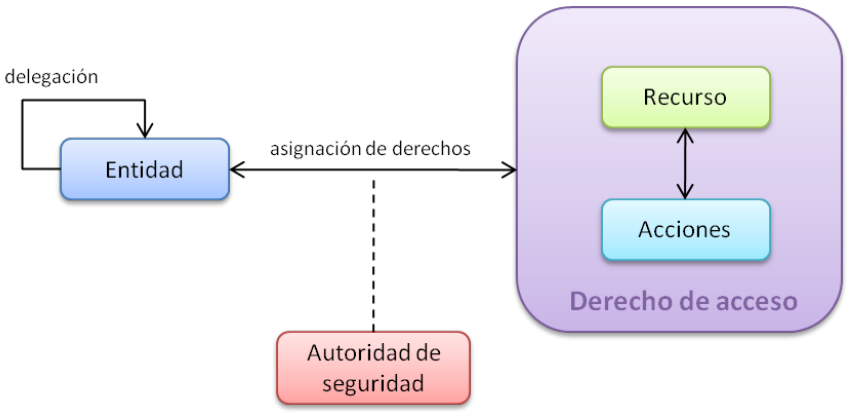

Figura 5.9. Esquema básico de control de acceso

Para completar el esquema se han incluido condiciones y obligaciones opcionales que pueden restringir los derechos de acceso y que son definidas por una autoridad de seguridad. Las condiciones determinarán bajo qué circunstancias es válido el derecho de acceso (si existe periodo de validez, si se tienen que tener en cuenta aspectos del entorno como hora del acceso o localización física de la entidad, etc.) y las obligaciones son acciones que deben realizarse antes, durante o después de un acceso permitido y su cumplimiento es obligatorio para que se complete el acceso al recurso. Algunos

ejemplos pueden ser el registro de la acción en un servicio de log o el envío de un mensaje de notificación a una autoridad de seguridad.

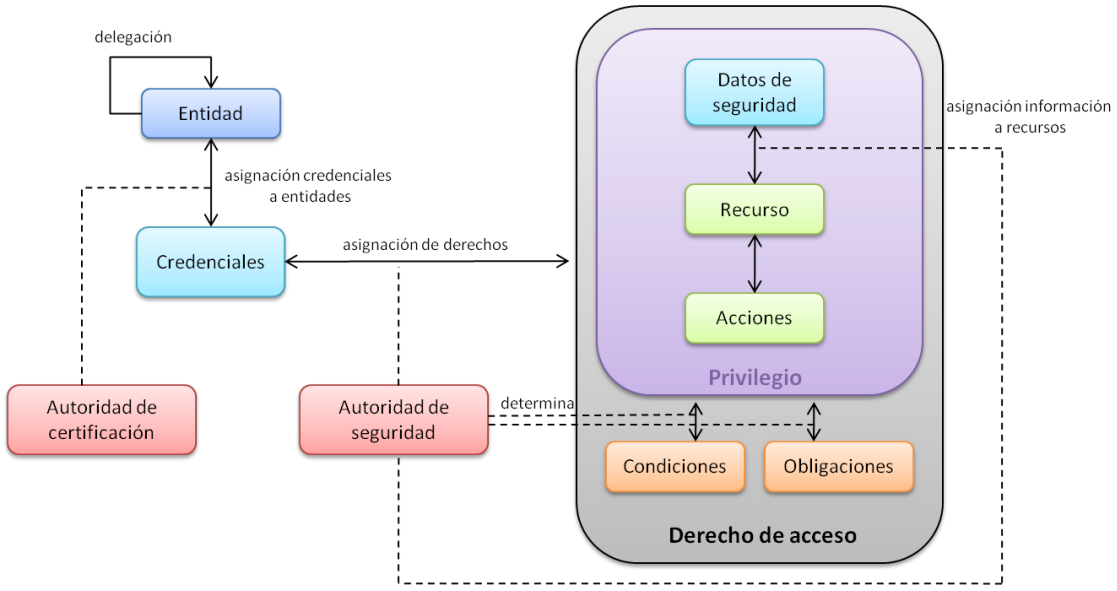

Figura 5.10. Esquema completo de control de acceso

Una vez definidas las nociones básicas y conceptos del control de acceso es necesario determinar los elementos lógicos que participan en un mecanismo de control de acceso. En la Figura 5.11 puede verse el esquema de operación de los mismos. En primer lugar una entidad tratará de acceder a un recurso y este intento tendrá que ser capturado por un elemento intermediario que hemos llamado de forma general *Gestor de acceso al recurso*. Este elemento derivará la petición a otro, el *Gestor de contexto*, encargado de reunir la información necesaria para tomar una decisión. Para ello hará uso de uno o varios *Proveedores de información* que contendrán datos de seguridad de recursos, credenciales, otros datos de entorno, etc.; los cuales podrán ser generales o estar dedicados a un tipo de información concreta. El gestor de contexto completa la solicitud de acceso con la información recuperada y se la pasa a un *Agente de decisión*, elemento encargado de juzgar el acceso y resolverlo. Para ello consultará un *Proveedor de políticas* que contendrá las reglas de políticas de seguridad del dominio, en general, y las que se aplican al recurso protegido en particular. Una vez tomada la decisión, y verificando que se cumplen las obligaciones (si las hubiera), el agente le devuelve la respuesta al gestor de contexto y éste a su vez al gestor de acceso al recurso. En caso de que se permita el acceso y existan obligaciones que cumplir se resolverán a través de un *Gestor de obligaciones*. Finalmente se le otorga a la entidad acceso al recurso protegido.

Una vez definido el marco de control de acceso, en la Tabla 5.I se establecen las correspondencias entre los elementos y conceptos definidos aquí y los que aparecen en las iniciativas descritas en el Capítulo 3. Sólo a través de la completa cobertura de estas iniciativas podemos afirmar que nuestro mecanismo de control de acceso es un marco armonizado que puede ser particularizado (a través de la nomenclatura) para los distintos esfuerzos. Como se desprende de la tabla, sólo el esquema de control de acceso de

XACML cubre todos los elementos y conceptos definidos en nuestro marco armonizado. El resto de iniciativas son completamente cubiertas si bien nuestro esquema es más general y se adapta a los distintos escenarios. Cabe destacar que no todos los elementos definidos en la Figura 5.11 tienen que estar presentes ya que los elementos son entidades lógicas que aglutinan funcionalidad específica de manera que un solo elemento físico podría condensar varios elementos lógicos.

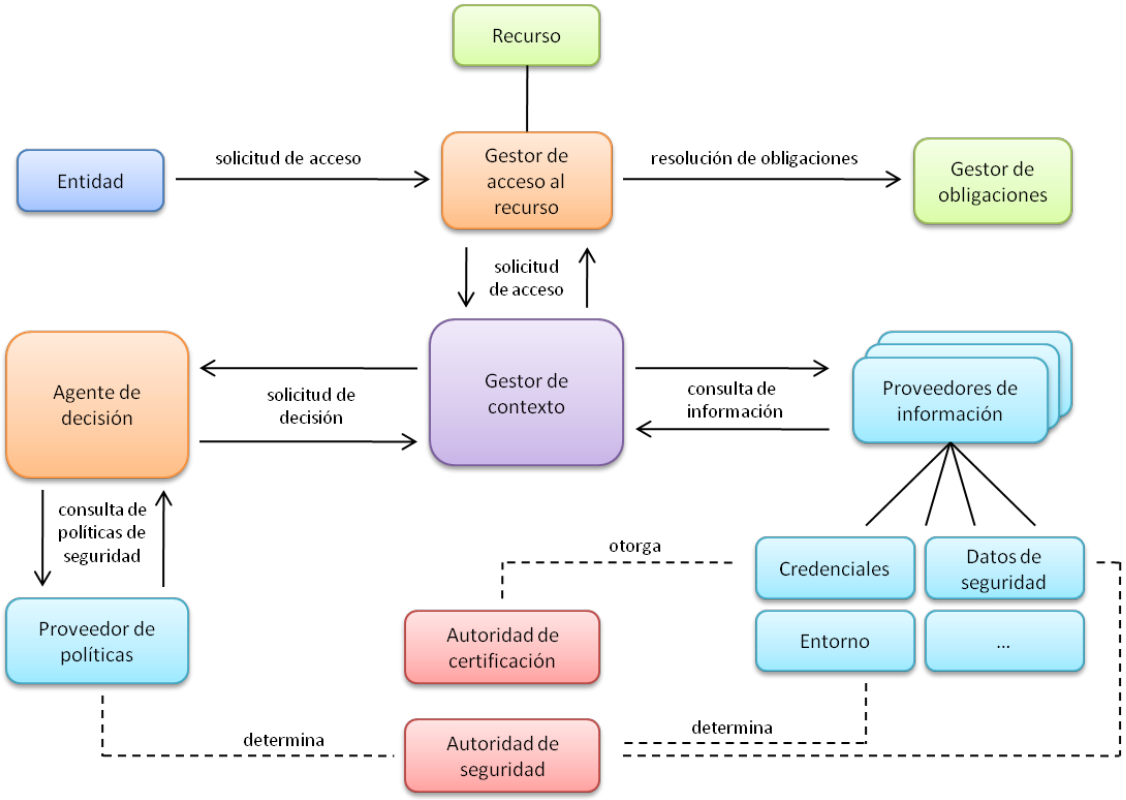

Figura 5.11. Esquema de elementos lógicos que forman parte del mecanismo de control de acceso

4. Resultado 3 - El sistema de infraestructura para el control de acceso

4.1. Introducción

Debido a la necesidad esencial de proteger los sistemas de usos malintencionados o ataques externos, la propuesta de incluir los aspectos que cubren las disciplinas de seguridad desde una etapa temprana en el diseño de los sistemas ha ido ganando empuje en los últimos años. Esto se debe principalmente a dos hechos. Por un lado, el desarrollo de un sistema con sus mecanismos de seguridad integrados desde la fase de diseño permite reducir la complejidad de las soluciones y simplificar las posteriores tareas de gestión, mantenimiento y evolución de los sistemas de seguridad. Por otro, la aparición de la ingeniería conducida por modelos (*Model-Driven Engineering*, MDE) como propuesta para formalizar todos los aspectos de los sistemas a través de modelos buscando cubrir con ellos todo el proceso de desarrollo desde las etapas de diseño hasta la implementación final. La especificación de requisitos de seguridad a través de modelos ha recibido importantes contribuciones en los últimos años. La dificultad que entraña

la especificación de requisitos de seguridad es que la amplia mayoría de las disciplinas presentan requisitos no funcionales como aspectos de integridad, confidencialidad, etc., que son difíciles de modelar.

Marco armonizado	ISO 10181-3	RBAC/ ABAC	XACML	IHE	HITSP	OMG RAD	ISO 22600
Derecho de acceso	Política control de acceso	Política control de acceso	Política XACML	---	Regla de políticas	Política	Política
Autoridad de seguridad	Autoridad del dominio	---	Autoridad de autorización	---	---	ADO	Fuente de autoridad
Autoridad de certificación	---	---	Autoridad de autenticación	---	---	Servicio de autenticación	
Credenciales	Acreditación	Rol/ Atributos	Atributos del sujeto	---	ACI del sujeto	---	Roles
Datos de seguridad	ACI del objetivo	Atributos	Atributos del recurso	---	ACI del objetivo	---	---
Gestor de acceso al recurso	AEF	---	PEP	PEP	PEP	Cliente RAD	Verificador
Gestor de contexto	---	---	Función de servicio de contexto	STS	---	---	
Agente de decisión	ADF	---	PDP	PDP	PDP	Evaluador de políticas	
Proveedor de políticas	---	---	PAP	Autoridad de políticas	---	Servicio de atributos dinámicos	---
Proveedor de información	---	---	PIP	Fuente de atributos	---	Servicio de atributos dinámicos	---
Gestor de obligaciones	---	---	Servicio de obligaciones	---	---	---	---

Tabla 5.I. Correspondencias entre los elementos definidos y los estándares del Capítulo 4

La autorización es, junto a la autenticación, una de las disciplinas que mejor permite la especificación en base a modelos, principalmente porque puede relacionarse más fácilmente con mecanismos concretos de implementación y los requisitos que plantea son más funcionales que en el caso de otras disciplinas. Las propuestas para el modelado del control de acceso que se pueden encontrar en la literatura son muchas y muy variadas. Algunas presentan lenguajes de modelado específicos y propietarios para abordar la seguridad [185] mientras que otras reutilizan o extienden herramientas de modelado ampliamente aceptadas (como es el caso de SecureUML que extiende UML [186]) o particularizan marcos de referencia como RM-ODP [187] o MDA [188] con aspectos de control de acceso.

En este apartado se presentan las contribuciones del presente trabajo de Tesis Doctoral en cuanto a la formalización del esquema normalizado de control de acceso del apartado anterior (Figuras 5.10 y 5.11) y su integración con la norma HISA (Apartado 2). Este esfuerzo pretende fijar los aspectos de diseño de la disciplina de control de acceso para la arquitectura sanitaria de referencia manteniéndose independiente de la tecnología (como dicta RM-ODP), por tanto se presentan aquí las tres vistas independientes de tecnología del sistema de control de acceso integrado en la arquitectura normalizada y posteriormente se particularizará para una tecnología concreta especificando los dos puntos de vista restantes.

4.2. La formalización del sistema de control de acceso y su integración en HISA

A continuación se presenta la formalización del mecanismo de control de acceso siguiendo la filosofía de RM-ODP, su lenguaje de empresa y la notación especificada por la norma ISO 19793 (UML4ODP). Este elemento forma parte de la arquitectura sanitaria basada en HISA y formalizada en el apartado anterior aunque la escasa cobertura que HISA realiza de aspectos de seguridad permite que su especificación se realice en su mayoría de manera independiente a ella. No obstante, por su importancia e interés, se destacan aquellos puntos concretos donde el componente de control de acceso se integra de forma coherente con la arquitectura.

4.2.1. Punto de vista de la Empresa

En el punto de vista de la Empresa se ha seguido el razonamiento establecido en el Apartado 2.2.1, esto es, se ha considerado que la comunidad HIS cubre toda la arquitectura sanitaria y, por tanto, debe incluir también todas las extensiones que se realicen a los fundamentos normativos de HISA. De esta forma se ha creado una nueva comunidad para acoger las capacidades de seguridad pero formando parte de la agregación de comunidades que es la comunidad HIS. Esta nueva comunidad se ha bautizado como *Security Infrastructure* (SI) y la decisión de hacerlo de esta manera ha surgido de la propia definición del concepto *comunidad*. De acuerdo con RM-ODP, una comunidad es una configuración de objetos modelando una colección de entidades (por ejemplo, personas, sistemas de procesamiento de información, recursos de varios tipos, etc.) que están sujetas a algún contrato explícito o implícito el cual gobierna su comportamiento colectivo y ha sido declarado para un objetivo particular. Por ello,

buscando promover la modularidad y escalabilidad de la formalización se ha decidido agrupar todos los aspectos relacionados con la seguridad en esta comunidad. Hay que decir que, aunque este trabajo se centra en la disciplina de autorización, cualquier disciplina de seguridad podría ser, y será en líneas futuras, incluida en esta comunidad. Por el momento el objetivo de la comunidad SI es gestionar privilegios de usuarios y atributos así como controlar el acceso a los recursos basándose en políticas. La Figura 5.12 muestra la especificación de la comunidad SI dentro de la agregación que es la comunidad HIS.

El mecanismo de autorización se basa en políticas de control de acceso que, a partir de atributos otorgados a usuarios y recursos, establecen los permisos, obligaciones y prohibiciones que rigen el acceso a estos últimos. Por ello, entre los procesos de empresa identificados (Figura 5.13) tenemos aquellos que gestionan los usuarios, los recursos y las políticas de control de acceso; aquellos procesos para la obtención de información de control de acceso; los de gestión de atributos; el proceso concreto de toma de decisión de acceso; y otros. A continuación se detallan cada uno de estos procesos, especificando qué roles de comportamiento están involucrados en los mismos y qué artefactos se manejan. En este punto es necesario mencionar que los procesos de gestión de usuarios y recursos tienen como campo de aplicación la seguridad del sistema y no la gestión real de estas entidades. Es decir, para que un usuario o recurso esté sujeto a las reglas de seguridad tendrá que estar registrado en la comunidad SI (lo cual se hace a través de los procesos *Register user* y *Register resource*). Como se verá a continuación existirán registros de usuarios y recursos para gestionar la información de cada tipo de entidad; así por ejemplo el proceso *Delete user registry* elimina el registro de un usuario concreto pero esto no quiere decir que se elimine al usuario del sistema completo. Obviamente, para alcanzar un escenario completamente seguro, será necesario que todos los recursos estén registrados en la comunidad SI y el acceso a ellos sujeto a políticas de control de acceso. Un usuario que no aparezca en los registros de seguridad no contará con información sobre él y, por tanto, su acceso a los recursos no será susceptible de ser permitido.

1. Registrar usuario (*Register user*): a través de este proceso se inscribe un usuario nuevo en la comunidad. Este proceso es iniciado por un actor con el rol de administrador de usuarios (*User Admin*) el cual se pone en contacto con el elemento que gestiona la información de los mismos y cuyo rol es el de registro de usuarios (*User Registry*). El primero pasa un artefacto con información básica sobre el usuario y solicita su registro en el sistema. Se realiza el proceso interno de inscripción y el registro devuelve una respuesta con el resultado del proceso. Si éste ha sido exitoso el administrador recupera el identificador del usuario en el sistema y termina el proceso.

2. Eliminar registro de usuario (*Delete user registry*): el proceso de empresa opuesto al anterior sirve para borrar el registro de un usuario en el sistema. Con los mismos roles involucrados ahora el administrador solicita la eliminación del registro del usuario pasando su identificador al registro de

usuarios. Éste realiza el proceso apropiado y devuelve la respuesta indicando el éxito o fracaso de la petición.

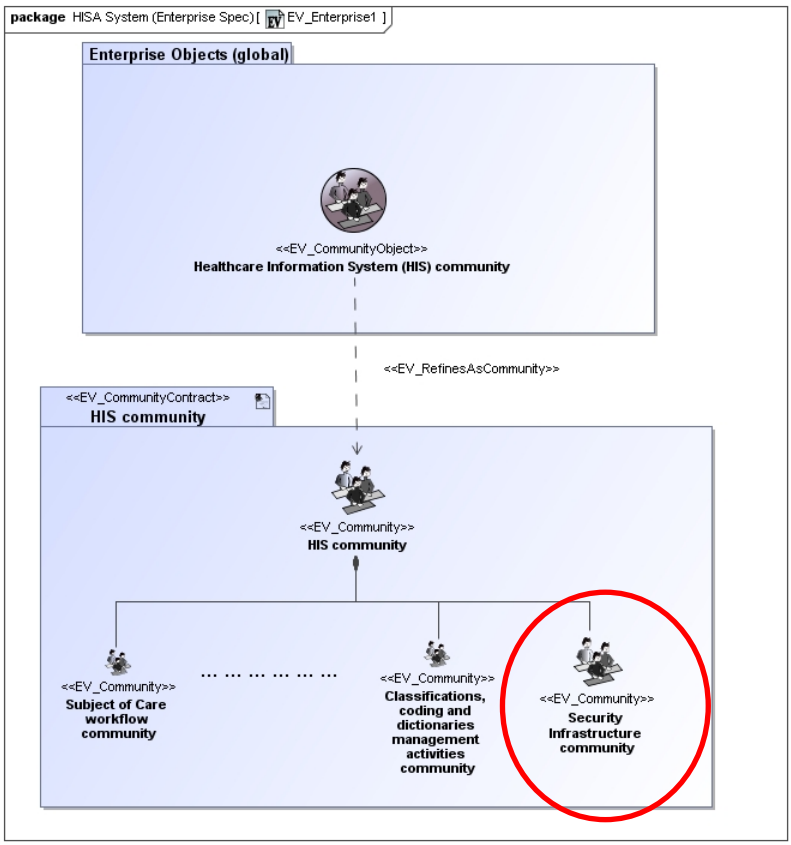

Figura 5.12. Punto de vista de la Empresa – comunidad de la infraestructura de seguridad

3. Eliminar registro de recurso (*Delete resource registry*): equivalente al anterior pero en este caso para eliminar el registro de un recurso. El administrador indica el recurso que solicita borrar a través de su identificador y recibe como respuesta el resultado del proceso de eliminación del registro.

4. Añadir política (*Add policy*): las políticas de control de acceso se tratan de manera similar a los registros de usuarios y recursos. Existe un elemento que desempeña el rol de proveedor de políticas (*Policy Provider*) el cual las almacena y gestiona ante la petición de otros actores. En este proceso es un actor con rol de administrador de políticas (*Policy Admin*) el que solicita la inclusión de una nueva política en el sistema. En la petición envía la política propiamente dicha y si el proveedor no encuentra problemas en su inclusión devuelve un identificador de la política en el sistema.

5. Editar política (*Edit policy*): el administrador de políticas puede solicitar al proveedor que edite una política ya incluida en el sistema, normalmente debido a que los términos que ésta rige (ya sean los usuarios, los recursos protegidos o las propias restricciones de acceso) han cambiado. El

administrador identifica la política que pretende editar y envía asimismo los detalles que deben modificarse. El proveedor de políticas actúa en consecuencia y devuelve el resultado del proceso.

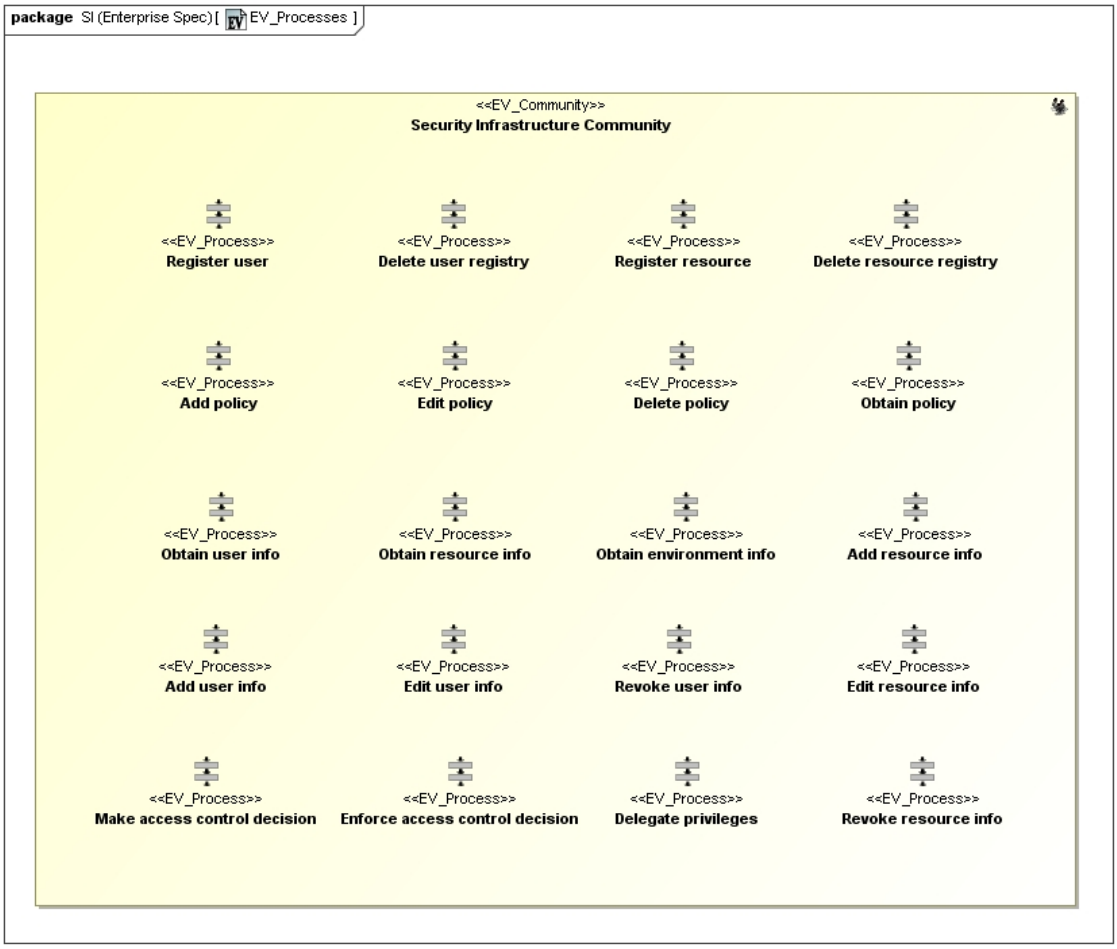

Figura 5.13. Punto de vista de la Empresa – procesos de la comunidad SI

6. Eliminar política (*Delete policy*): el proceso necesario para que el administrador de políticas solicite la eliminación de una política concreta a través del identificador de la misma. El proveedor realizará las comprobaciones apropiadas y devolverá el resultado del proceso.

7. Obtener política (*Obtain policy*): al margen de los procesos de gestión que puede realizar el administrador de políticas, cualquier otro elemento del sistema con capacidad para desempeñar el rol de solicitante de políticas (*Policy Requester*) puede interaccionar con el proveedor de políticas. A través de este proceso el solicitante envía cierta información sobre la política (puede ser su identificador, algún campo de la misma, etc.) y el proveedor devuelve aquella (o aquellas) política que encaja con la información suministrada. Este proceso es esencial en la toma de decisiones puesto que, ante un intento de acceso, el elemento responsable de controlarlo tendrá que recuperar aquellas políticas aplicables al caso concreto que se dé ya sea buscando las políticas referentes al usuario que pretende acceder, el recurso accedido u otras condiciones del entorno.

8. Recuperar información de usuario (*Obtain user info*): existe un rol, *Information Requester*, que pueden desempeñar multitud de elementos y que permite solicitar información sobre usuarios del sistema. A través de este proceso un elemento con este rol indica el identificador del usuario sobre el que quiere recuperar la información al registro correspondiente y éste devuelve toda la información almacenada (en concreto, atributos) sobre el usuario. Este proceso forma parte de la toma de decisiones puesto que, ante un usuario accediendo a un recuso, se necesitará recuperar toda la información sobre el mismo que exista en el sistema y que facilite la toma de decisión de acceso.

9. Recuperar información de recurso (*Obtain resource info*): de forma similar al proceso anterior, un solicitante de información puede consultar los atributos almacenados en un registro del sistema sobre cierto recurso. Este proceso asiste de igual forma que el anterior a la toma de decisiones de acceso.

10. Recuperar información del entorno (*Obtain environment info*): en general el proceso de toma de decisiones no se basa exclusivamente en la información sobre los usuarios y recursos sino que incluye condiciones del entorno que pueden ser muy heterogéneas (por ejemplo, fecha y hora del acceso, localización física desde la que se accede, etc.). A esta información se le ha venido a llamar información del entorno y un elemento desempeñando el rol de solicitante de información puede recuperarla desde el proveedor correspondiente (*Environment Information Provider*) análogamente a como se hace en los procesos anteriores para la información de usuarios y recursos.

11. Añadir información de recurso (*Add resource info*): para todos los recursos registrados en el sistema el administrador correspondiente puede incluir cierta información (en nuestro caso, atributos) que facilitará las tareas de toma de decisiones de acceso. A través de este proceso el administrador de recursos solicita la inclusión de esta información indicando en la petición el identificador del recurso al que se refiere. El registro de recursos realizará las tareas correspondientes y responderá con el resultado de las mismas. Este proceso se lleva a cabo cuando no existía información previa asociada al registro de un recurso ya que en caso contrario se ejecutaría el proceso número 12.

12. Editar información de recurso (*Edit resource info*): si, aparte del registro, ya existe información sobre un recurso en el sistema, en lugar del proceso anterior se utiliza este para añadir nueva información o modificar la existente. De forma análoga al proceso 11, el administrador de recursos pasa la información a añadir junto con el identificador del recurso al registro de recursos.

13. Revocar información de recurso (*Revoke resource info*): para eliminar toda (o parte de) la información que existe sobre un recurso concreto el administrador de recursos solicita la

revocación al registro de recursos identificando el elemento concreto del cual quiere borrar la información.

14. Añadir información de usuario (*Add user info*): al igual que pasa con los recursos, para todos los usuarios incluidos en el sistema el administrador de usuarios debe poder incluir junto a los registros de los mismos cierta información (en nuestro caso, atributos) que faciliten las tareas de toma de decisiones de acceso. A través de este proceso el administrador de usuarios solicita la inclusión de esta información indicando en la petición el identificador del usuario al que se refiere. El registro de usuarios realiza las tareas correspondientes y responde con el resultado de las mismas. Este proceso se lleva a cabo cuando, aparte del registro de usuario, no existe información sobre el mismo.

15. Editar información de usuario (*Edit user info*): si ya existe información sobre un usuario en el sistema, en lugar del proceso anterior se utiliza éste para añadir nueva información o modificar la existente. De forma análoga, el administrador de usuarios pasa la información a añadir junto con el identificador del usuario al registro de usuarios.

16. Revocar información de usuario (*Revoke user info*): para eliminar toda (o parte de) la información que existe sobre un usuario concreto el administrador de usuarios solicita la revocación al registro de usuarios identificando el elemento concreto del cual quiere borrar la información.

17. Toma de decisión de control de acceso (*Make access control decision*): este es el proceso más complejo de la comunidad SI y en él están incluidos otros procesos vistos anteriormente. Cuatro roles de comportamiento comparten las acciones del proceso: un solicitante de acceso (*Access Requester*), un gestor de acceso a recurso (*Resource Access Manager*), un gestor de contexto (*Context Manager*) y un agente de decisión (*Decision Agent*). El solicitante de acceso (que como se verá más adelante puede ser una persona o un sistema representando a una) intenta acceder a un recurso concreto y es el gestor de acceso a ese recurso el que compone la petición de acceso indicando el identificador del usuario y del recurso protegido. Esta petición llega al gestor de contexto que dispara los procesos de recuperación de información sobre el usuario, el recurso y el contexto. Una vez recogida toda la información necesaria, el gestor de contexto compone la petición de decisión y se la envía al agente de decisión. Éste realiza el proceso de decisión interno y resuelve autorizando el acceso (*grant*) o denegándolo (*deny*). Se contemplan otras posibilidades (por ejemplo, información insuficiente) con el tercer tipo de respuesta (*other*). El gestor de contexto devuelve la respuesta al gestor de acceso al recurso y éste dispara el proceso de cumplimiento de la decisión de acceso (ver proceso número 19).

18. Cumplimiento de la decisión de control de acceso (*Enforce access control decision*): una vez que se ha tomado una decisión, el gestor de acceso al recurso es el encargado de cumplir con lo que se haya resuelto en dicha decisión. Si el acceso ha sido denegado (o en otros supuestos

indeterminados) se comunica la decisión al solicitante del acceso y se termina puesto que el acceso no ha sido autorizado. En caso positivo, se recuperan las obligaciones incluidas en la autorización de acceso y se envían al elemento que desempeñe el rol de gestor de obligaciones. Éste las resuelve y responde al gestor de acceso al recurso con el resultado. Si no ha habido ningún problema con las obligaciones, el acceso se permite al solicitante.

19. Delegación de privilegios (*Delegate privileges*): este último proceso recoge el mecanismo de delegación de privilegios llevado a cabo por un elemento o usuario desempeñando el rol de solicitante de delegación. Éste elemento envía al registro de usuario las credenciales que le autorizan para la delegación e información sobre el usuario delegado. El registro actualiza la información de los usuarios y devuelve el resultado de este proceso interno.

A partir de los procesos identificados se pueden extraer los distintos roles de comportamiento que se dan en la comunidad SI, los cuales se pueden clasificar en:

- Roles de usuarios: *Access Requester*, *Policy Requester*, *Information Requester* y *Delegation Requester.*

- Roles de administradores: *User Admin, Policy Admin* y *Resource Admin.*

- Roles de proveedores de información: *User Registry, Resource Registry, Environment Information Provider* y *Policy Provider.*

- Roles de intermediarios del proceso de control de acceso: *Resource Access Manager, Decision Agent, Context Manager* y *Obligation Manager.*

En la identificación de los objetos de empresa se tienen: individuos (*Admin*, *Resource Owner* y *User),* objetos (*Resource*, *User Info*, *Resource Info* y *Access Control Policy)* y sistemas (*Policy Information System, User Information System, Environment Information System, Resource Information System, Context Handler System, Obligation System, User Agent, Policy Enforcement System* y *Policy Decision System*).

Hay que destacar que en la relación entre actores y roles se ha escogido el escenario simple donde se cuenta con un sistema de cada tipo. Obviamente la complejidad del escenario real puede ser mayor si se incluyen, por ejemplo, varios dominios organizacionales. Cómo se resuelve la distribución de elementos no es un aspecto a considerar en el punto de vista de la Empresa (se hará en el de Ingeniería) pero es necesario destacar que en un escenario que implique múltiples dominios administrativos es muy probable contar con varias instancias de los diferentes sistemas. Así, por ejemplo, un sistema gestor de contexto en un dominio (*Context Handler System*) podría recibir una petición de acceso que no pudiesen resolver los sistemas de decisión de políticas (*Policy Decision System*) accesibles por él. En ese caso la petición debería enviarse a otro gestor de contexto perteneciente a otro dominio. En ese caso el primer

gestor tomaría el rol de gestor de acceso al recurso para el segundo gestor de contexto. En líneas generales y para resumir, se considera que cualquier componente puede contactar con sus iguales en otros dominios en caso de no poder realizar su tarea específica en el sistema de control de acceso, desempeñando un rol distinto al que originariamente tiene asignado. De esta forma se convertiría en usuario de componentes análogos a él en otros dominios. En la mayoría de los casos los agentes que podrían presentar este comportamiento adicional serán los sistemas de información o registros (de políticas, usuarios, entorno o recursos), el sistema de obligaciones y el de decisión de políticas.

Para terminar con la especificación del punto de vista de la Empresa de la comunidad SI se presentan a continuación las políticas que limitan o restringen el comportamiento de los objetos de empresa y roles:

- *Resource access manager*: especifica que sólo un gestor de acceso puede estar asignado en un momento dado al control de acceso a cada recurso de manera que no haya diferentes elementos intermediarios para acceder a un recurso concreto. Esta política influye en los procesos de "añadir recurso", "toma de decisión de acceso" y "cumplimiento de decisión de control de acceso". Excepcionalmente un gestor de acceso puede estar redundado para satisfacer requisitos de fiabilidad y eficiencia. En ese caso no se consideran dos intermediarios separados sino como el mismo elemento duplicado pero manteniendo la coherencia entre sus partes y, por tanto, se sigue cumpliendo la política.

- *User information*: establece que la información de usuario puede estar distribuida pero manteniendo la coherencia entre sus partes, que poseerá siempre un periodo de validez y que cada pieza de información contendrá datos sobre su fecha de creación, autoría, gestión de cambios, versiones, etc. Esta política debe tenerse en cuenta en todos los procesos que involucran información de usuario.

- *Resource information*: esta política es análoga a la anterior pero para la información de los recursos.

- *Access control policies*: las políticas de control de acceso también están restringidas según especifica esta política de comportamiento. Por un lado, deben tener una serie de campos como un identificador único, un periodo de validez y datos sobre fecha de creación y autor, control de versiones, etc. Por otro lado, tendrán que existir mecanismos o reglas que permitan establecer prioridades entre las políticas y resolver conflictos.

- *Resource identifier unicity* y *User identifier unicity*: estas políticas especifican que cada recurso y usuario debe tener un identificador único dentro del sistema.

- *Obligations*: esta política introduce las obligaciones dentro de la toma de decisiones de control de acceso. En concreto determina que una decisión favorable de acceso a un recurso puede estar

condicionada por cero o más reglas de obligación. En caso de que existan reglas de obligación, satisfacerlas será condición indispensable para otorgar el acceso.

- *Delegations*: el mecanismo de delegación de privilegios es ajustado mediante esta política. En concreto establece que un usuario será capaz de delegar todos o parte de sus privilegios a otro usuario. Además, un elemento con el rol de Agente de Usuario obtiene automáticamente los privilegios del usuario al que representa.

- *Access control*: en esta política se indica que el control de acceso a un recurso debe ser determinado por una o más políticas de control de acceso. En caso de que sea imposible tomar una decisión sobre el acceso a un recurso (no existen políticas o suficiente información), se opta por la decisión más conservativa, esto es, el acceso se deniega.

- *Least privileges*: esta política indica que los privilegios serán otorgados a los usuarios en función del principio de menor privilegio y el propósito de uso del recurso.

- *Access control policy groups*. Por último, se indican los dos grupos de políticas de control de acceso que deben aparecer como mínimo en el sistema, estos son: el grupo de políticas que siguen las preferencias y el consentimiento del ciudadano, y el grupo de políticas por defecto (ya sean legislativas o necesarias para casos excepcionales de emergencia).

4.2.2. Punto de vista de la Información

El punto de vista de la Información de la infraestructura de control de acceso se formaliza a través de cinco esquemas invariantes. Aunque están relacionados entre sí se ha optado por separar los objetos de la vista de la Información en cinco grupos de acuerdo a su función dentro de los procesos de control de acceso. Así tenemos objetos relacionados con las propias políticas de control de acceso (*Access control policy objects*), la información que se maneja en esas políticas sobre los recursos, usuarios y contexto (*Access control information objects*), los usuarios y agentes involucrados en el control de acceso (*Users and agents objects*), los mensajes que se intercambian en los procesos de la infraestructura (*Access control messaging objects*) y las acciones que se pueden realizar sobre los objetos de información (*Information actions*).

Antes de describir los objetos de información incluidos en la infraestructura de control de acceso, y como la comunidad SI forma parte de la comunidad HIS, merece especial atención destacar aquellos aspectos del punto de vista de la Información de HISA que tienen que ver con la autorización y cómo encaja la presente propuesta con ellos. Por un lado, lo que especifica el estándar ISO12967 sobre control de acceso es que las autorizaciones se basan en perfiles de autorización (*Authorization profiles*) que relacionan usuarios (*User* y *Agent*) con recursos controlados (*Controlled element*). La propuesta presentada en esta Tesis Doctoral encaja con estos conceptos y extiende las posibilidades añadiendo mayor sofisticación y concreción a los posibles mecanismos de control de acceso. En los modelos del

punto de vista de la Información aquí descrito los objetos pertenecientes a la norma HISA han sido representados por el círculo con la 'i' de información mientras que los propios de esta especificación aparecen como cuadrados con sus atributos incluidos. Esto no se ha realizado en el punto de vista de la Empresa porque HISA no especifica ningún objeto de ese punto de vista para el conjunto de actividades de usuarios y autorizaciones. Otro aspecto a mencionar con respecto a las autorizaciones en HISA tiene que ver con los atributos que la norma especifica para todos los objetos de información de cualquier arquitectura sanitaria conforme a ella. Entre los atributos obligatorios existe un conjunto llamado de "atributos del sistema" (*SystemAttributes*) comunes a todos los objetos de información y que satisfacen requisitos generales para registro, auditoría, etc., de cada instancia. En este grupo se encuentra un atributo llamado "autorización" (*authorization*) que toma como tipo de dato la cadena de caracteres. Su definición establece que especifica restricciones con respecto a los derechos de autorización sobre la lectura, actualización o borrado de la instancia específica. Aunque este atributo sirve como mecanismo básico de autorización, su alcance está limitado a controlar el acceso sobre las instancias de objetos de información, no pudiendo extrapolarse a recursos físicos o lógicos (que también son objeto de nuestra infraestructura de control de acceso). En lo sucesivo consideraremos el uso de este atributo para la autorización de las instancias de los objetos de información almacenando en la cadena del atributo, por ejemplo, el identificador de la política de control de acceso aplicable en cada caso. Este mecanismo normativo de HISA no se solapa con la extensión en seguridad que se realiza en este trabajo teniendo en cuenta que tienen dos niveles de aplicación diferenciados. El normativo de HISA se aplica a las instancias de objetos de información y el de esta Tesis, además de a esas mismas instancias, se podrá aplicar a otros recursos físicos o lógicos.

A continuación se describen los principales objetos de información incluidos en la infraestructura de control de acceso.

1. Objetos de información para el control de acceso (*Access control information objects*, Figura 5.14)

Los objetos relacionados con la información necesaria para el control de acceso están unidos a la especificación de la norma HISA a través de los objetos de información *User* y *Controlled element.* Cada usuario y elemento controlado (al que nos referiremos como recurso protegido o simplemente recurso) puede tener asignado un conjunto de piezas de información que corresponden con los objetos *User info* y *Controlled element info.* Por otro lado, la información sobre usuarios y recursos será determinada por una autoridad (objeto de información *Authority*) que incluye un identificador y un nombre. Finalmente, siguiendo el modelo de control de acceso armonizado propuesto, las piezas de información sobre usuarios y recursos estarán compuestas de atributos que contendrán un identificador, un nombre y un valor y podrán ser de muchos tipos. En el modelo de información de la Figura 5.14 se muestran varios ejemplos como localización (*location*), rol (*role*) y propósito de uso (*purpose of use*).

2. Objetos de información de políticas de control de acceso (*Access control policy objects*, Figura 5.15)

El objeto de información principal de este grupo es el referido a una política (*Policy*) el cual hace de nexo de unión con la norma HISA puesto que se identifica como un heredero del objeto perfil de autorización (*Authorization profile*). Se especifica aquí la existencia de dominios de seguridad (*Domain*) en los cuales operarán autoridades (*Authority*) creando y manteniendo políticas de control de acceso. Tres estándares han sido utilizados para el resto de objetos de información:

- ISO/TS 22600 (Gestión de privilegios y control de acceso): esta norma introduce principios y especifica servicios necesarios para la gestión de privilegios y el control de acceso en sistemas de información sanitarios, poniendo especial interés en las colaboraciones entre organizaciones a través de fronteras administrativas y de seguridad. De los modelos formales que aporta se identifican, además del objeto *Domain*, los objetos de información referidos a tipos de políticas como son *Basic policy*, *Composite policy*, *Refrain policy*, *Authorization policy*, *Auth+*, *Auth-*, *Obligation policy*, *Delegation policy*, *Deleg+* y *Deleg-*. La política básica relaciona a un usuario (objeto de información de HISA *User*) con un recurso (objeto *Controlled element* de la misma norma). La Figura 3.14 mostró el diagrama con el que la norma ISO 22600 presenta su modelo formal de políticas.

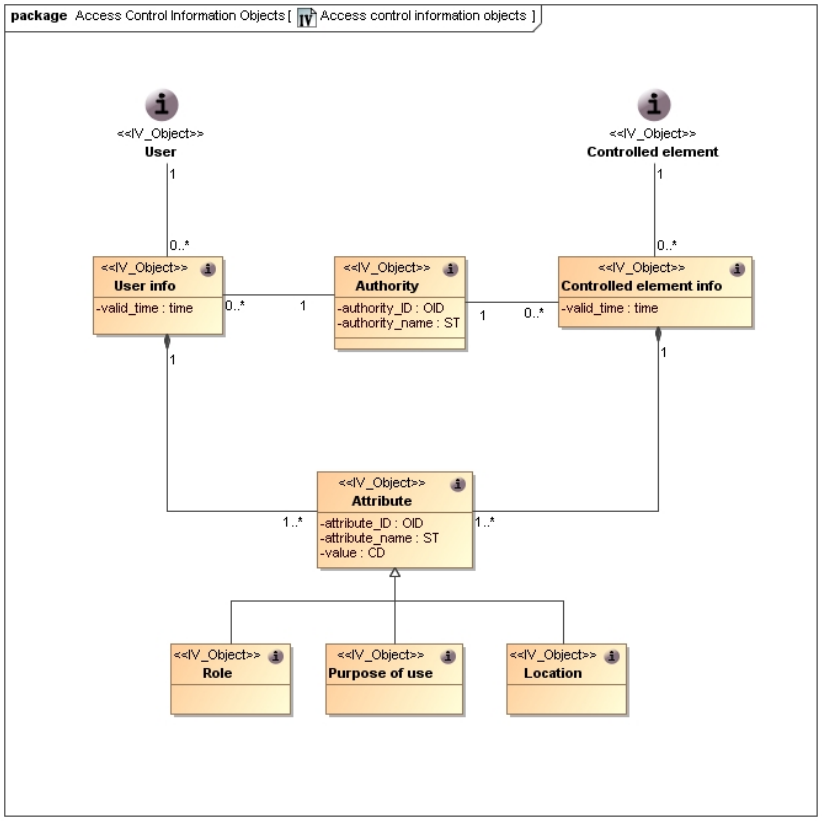

Figura 5.14. Modelo de información para los objetos de información de control de acceso

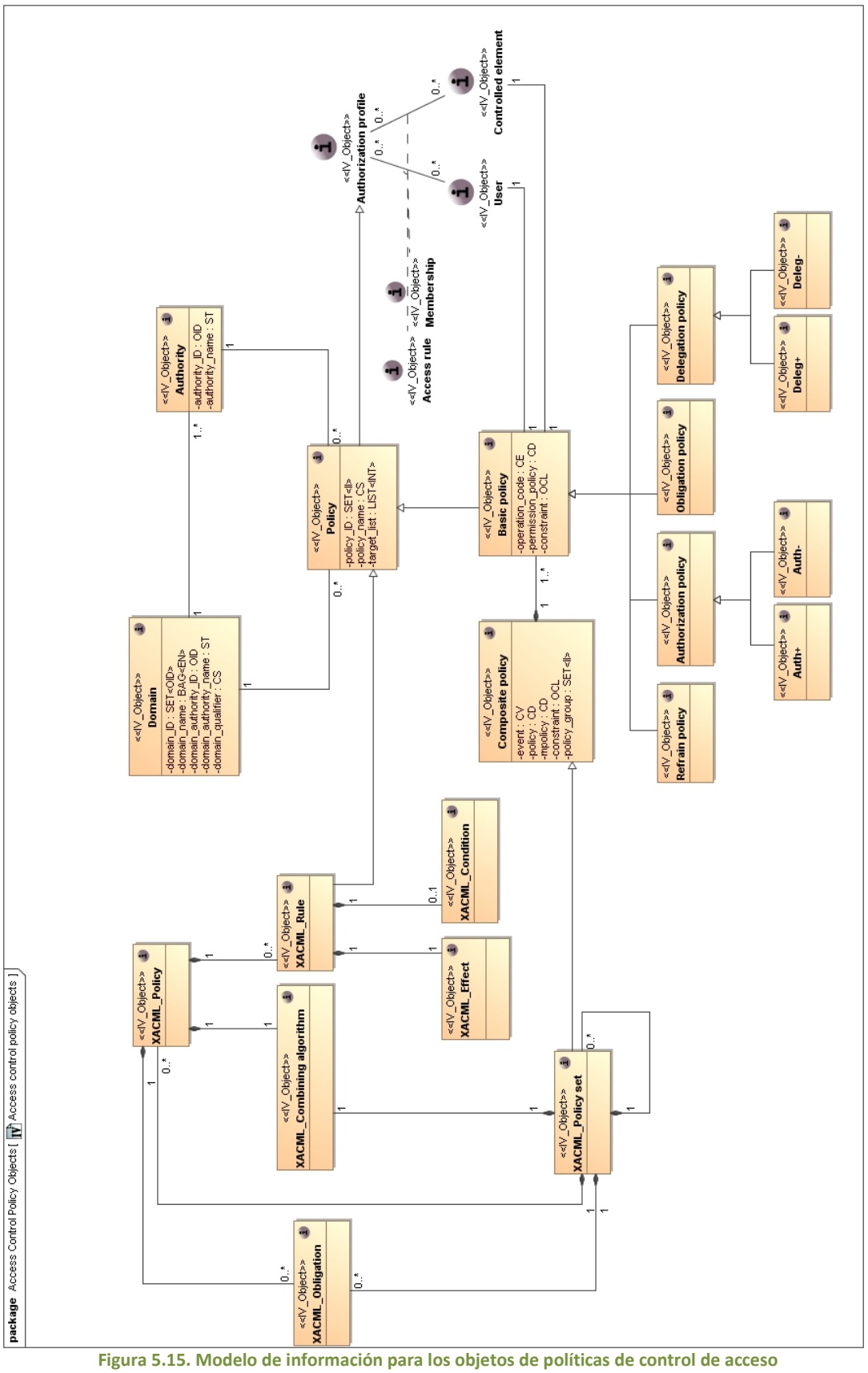

Figura 5.15. Modelo de información para los objetos de políticas de control de acceso

- ITU-T X.1142 (XACML): determinaba un lenguaje de definición de políticas y su estructura se ha llevado a objetos de información en la propuesta de este trabajo. En concreto todos los objetos

con prefijo XACML han sido heredados de la norma. Tal y como se establece en el estándar, una política completa aplicable a una determinada petición de decisión puede estar compuesta por varias reglas o políticas individuales. XACML define los tres elementos de política más generales: <*Rule*> (en el modelo representado por el objeto *XACML_Rule*), <*Policy*> (*XACML_Policy*) y <*PolicySet*> (*XACML_Policy set*). El elemento <*Policy*> contiene un conjunto de elementos <*Rule*> y un procedimiento especificado para combinar los resultados de su evaluación. Es la unidad de política básica y, por lo tanto, se considera que es la base de una decisión de autorización. El elemento <*PolicySet*> contiene un conjunto de elementos <*Policy*> u otros <*PolicySet*> y un procedimiento especificado para combinar los resultados de su evaluación. Es el medio normalizado para combinar políticas separadas en una sola política combinada. Debido a que el modelo de la norma ISO 22600 es más general y de mayor alcance que la norma XACML, los objetos pertenecientes a ésta última se han relacionado con la primera a través de relaciones de herencia. Así, una *XACML_Rule* hereda del objeto general *Policy* al igual que lo hace el objeto *Basic policy*, mientras que un *XACML_Policy set* es una especialización de una *Composite policy.*

- Recomendaciones de seguridad de HL7: dentro del amplio conjunto de estándares desarrollado por el grupo HL7, también la seguridad y el control de acceso han sido el foco de algunos esfuerzos y recomendaciones. Debido a que la propuesta de HL7 para el control de acceso sigue la filosofía RBAC, el conjunto de objetos de información relacionado con los usuarios es diferente a la propuesta que se sigue en este trabajo. Donde sí se solapan ambos modelos de información es en el conjunto de objetos relacionados con las políticas. Así tenemos los objetos: *Policy, Basic Policy, Composite Policy, Delegation Policy, Authorization Policy, Obligation Policy, Refrain Policy* y *Authority.*

3. Objetos de información de usuarios y agentes (*Users and agents objects*, Figura 5.16)

Este paquete de objetos de información organiza aquellos usuarios y agentes que toman parte en los procesos de control de acceso. Por un lado, el elemento autoridad (*Authority*) que ya apareció anteriormente ahora se relaciona con un conjunto de administradores de seguridad (*Security admin*) que se referirán a personas físicas (por eso hereda de la clase *Person* de la norma HISA). Además se incluye el objeto de información sobre las personas propietarias de los recursos (*Resource owner*) que, como se vio en el punto de vista de la Empresa, resultan de importancia ya que pueden tomar el rol de administrador de recursos. Por otro lado, en este paquete se incluyen también aquellos objetos de información que se refieren a los diferentes sistemas o componentes de la arquitectura que toman parte del control de acceso. Así tenemos el sistema de decisión de políticas (*Policy decision system*), de obligaciones (*Obligation system*), el gestor de contexto (*Context handler system*), el de cumplimiento de políticas (*Policy enforcement system*) y los de información (*Information system*) que se especializan en de información de entorno (*Environment information system*), de recursos (*Resource information system*), de usuarios (*User information system*) y de políticas (*Policy information system*). El objeto de información *Policy enforcement system* puede estar relacionado con recursos (*Controlled element*) pero

aquí se satisface la política de empresa en la que se especificaba que todos los recursos deben estar gestionados por algún sistema que controle el acceso. Finalmente, los sistemas de información estarán relacionados con los objetos de información que les corresponden, esto es, el objeto *User information system* podrá relacionarse con piezas de información de usuario (*User info*) al igual que lo hacen el de recursos con información de los mismos y el sistema de políticas con éstas.

4. Objetos de información de mensajes (*Access control messaging objects*, Figura 5.17)

En este paquete de objetos de información se recogen aquellos relacionados con los mensajes que se intercambian en la toma de decisiones de control de acceso. Por un lado, un solicitante de acceso (*Access requester*), el cual hereda del objeto *User* (es decir, puede ser una persona o un sistema), compone una petición de acceso (*Access request*) a un recurso (*Controlled element*) para realizar una operación concreta (*Operation*). Un sistema de decisión recibirá esta petición y consentirá o denegará el acceso. La respuesta (*Access control decision*) estará relacionada uno a uno con la petición y podrá contener obligaciones (*Obligation policy*) que deben resolverse y condiciones *(XACML_Condition*) que deben cumplirse antes de otorgar el acceso.

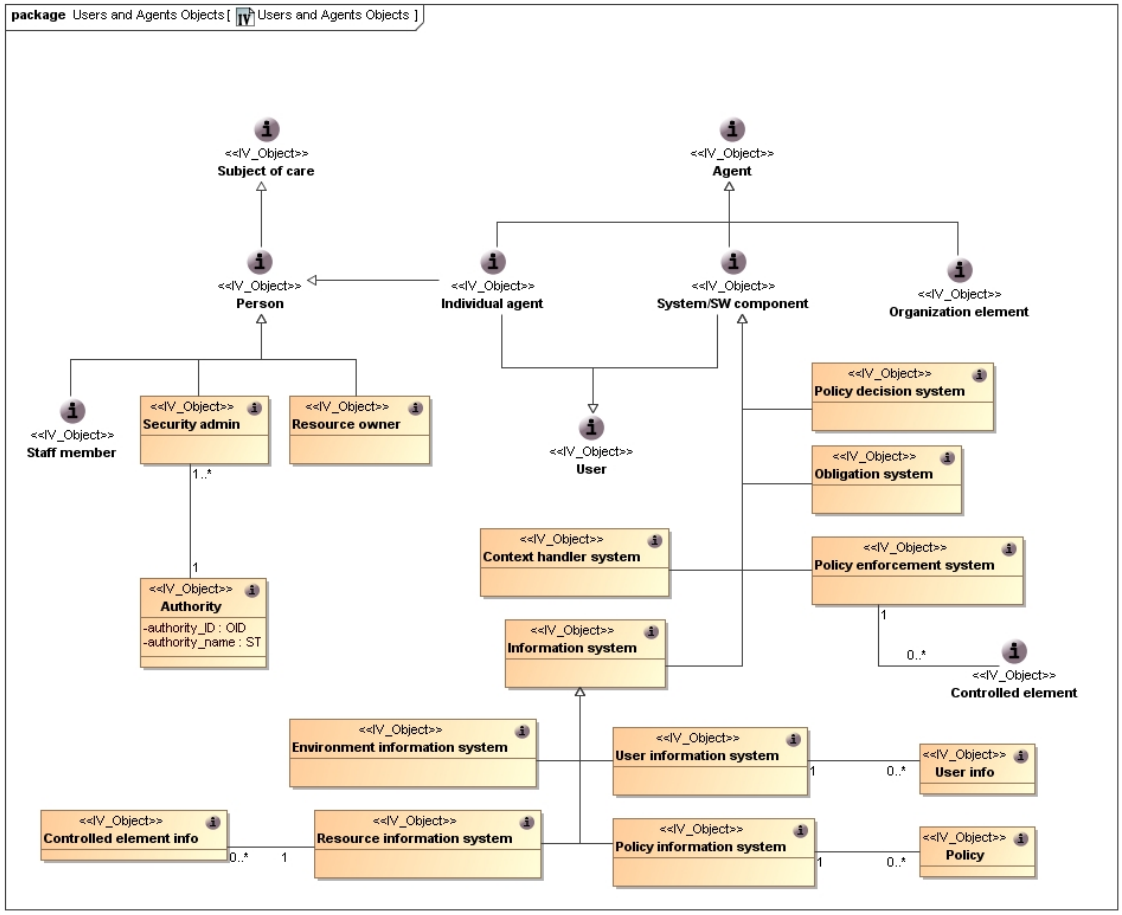

Figura 5.16. Modelo de información para los objetos de agentes y usuarios

5. Acciones de información (*Information actions*)

El quinto esquema invariante que forma parte de la especificación del punto de vista de la Información es el que agrupa las acciones de información. Éstas se definen como aquellas operaciones del sistema en las que participa al menos un objeto de información el cual, generalmente, cambia su estado con motivo de la ejecución de dicha acción. A partir de los procesos e interacciones definidos en la vista de la Empresa se han identificado las acciones de información. Tenemos acciones relacionadas con los usuarios (*addUser* y *deleteUser*), con los recursos (*addResource* y *deleteResource*), las políticas (*addPolicy, editPolicy* y *deletePolicy*), la información de usuarios (*addUserInfo*, *editUserInfo* y *revokeUserInfo*) y recursos (*addResourceInfo*, *editResourceInfo* y *revokeResourceInfo*), los atributos (*addAttribute* y *deleteAttribute*), la petición (*composeRequest*) y la decisión de acceso (*makeDecision* y *processAccessControlDecision*).

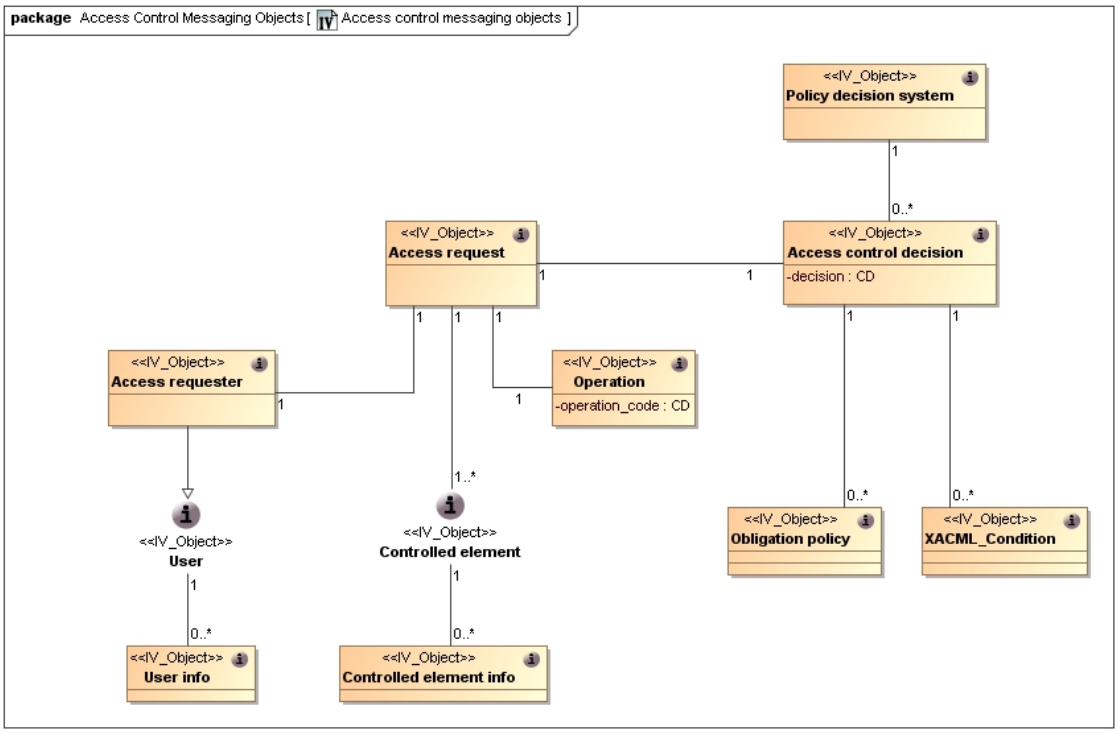

Figura 5.17. Modelo de información para los objetos de mensajes de control de acceso

4.2.3. Correspondencia entre los puntos de vista de la Empresa y la Información

Las correspondencias entre los puntos de vista de la Empresa y la Información para el sistema de control de acceso están contenidas en tres paquetes. El primero recoge las correspondencias entre los objetos que en ambos puntos de vista tienen el mismo nombre como *Policy information system*, *User information system*, *Environment information system*, *Resource information system*, *Context handler system*, *Policy information system*, *Obligation system*, *Policy decision system*, *Person* y *Resource owner.* También se incluyen objetos que, aun teniendo nombres diferentes en cada punto de vista, se refieren al mismo elemento del sistema (*Admin* con *Security admin* y *User agent* con *Individual agent*).

El segundo paquete también contempla relaciones entre objetos de empresa e información pero incluye además aquellos procesos de empresa y acciones de información que se relacionan con ellos. De esta forma se especifican las siguientes correspondencias:

- Entre los objetos *User* (ambos homónimos) que cuentan con los procesos *Register user* y *Delete user registry* y las correspondientes acciones de información *addUser* y *deleteUser.*

- Se establece el mismo tipo de correspondencia entre el objeto de empresa *Resource* y el de información *Controlled element.* Los procesos y acciones son *Register resource* con *addResource* y *Delete resource registry* con *deleteResource.*

- El objeto de empresa *Access control policy* se corresponde con el objeto de información *Policy* y tienen tres procesos y acciones relacionados: *Include policy* con *addPolicy, Edit policy* con *editPolicy* y *Delete policy* con *deletePolicy.*

- Por último se tiene la relación entre los objetos de empresa e información *User info* cuyos procesos y acciones son *Add user info* con *addUserInfo, Edit user info* con *editUserInfo* y *Revoke user info* con *revokeUserInfo.*

El último paquete posee la correspondencia entre el objeto de empresa *Resource info* y el de información *Controlled element info.* Los procesos de empresa y acciones de información asociados son: *Add resource info* con *addResourceInfo, Edit resource info* con *editResourceInfo* y *Revoke resource info* con *revokeResourceInfo.* Al margen de esto se establecen además las correspondencias entre los procesos *Make decision* y *Compose request* con las acciones *makeDecision* y *composeRequest* respectivamente.

4.2.4. Punto de vista Computacional

El punto de vista Computacional aporta a la descripción del sistema la descomposición funcional del mismo de forma transparente a la distribución. La especificación computacional define unidades de función como objetos computacionales además de las interacciones entre ellos sin considerar la distribución a través de redes o nodos. Esta especificación se divide en cuatro paquetes:

- Plantillas de objetos (*Object templates*) y plantillas de interfaces (*Interface templates*): presentan los objetos computacionales y las interfaces que poseen,

- Tipos de datos (*Data types*): muestra los tipos de datos que utilizan los objetos computacionales, y

- Firmas de interfaces (*Interface signatures*): recoge las interfaces que exponen los objetos computacionales con los métodos que aúnan cada una de ellas.

Para facilitar el entendimiento de la especificación del punto de vista Computacional se va a exponer en primer lugar cómo se alcanza la conformidad de la misma con la norma HISA. El requisito normativo indispensable es que exista un objeto computacional por cada clase identificada en el punto de vista de la Información. Cada uno de estos objetos computacionales básicos deberá proveer con el conjunto de métodos básicos definidos en la norma. Con todo este conjunto de objetos computacionales HISA resuelve la gestión de todas las clases definidas en el modelo de información así como las propiedades e instancias de cada una de ellas. Paralelamente a este grupo de objetos computacionales básicos, el sistema de control de acceso propiamente dicho se descompone funcionalmente en seis elementos (Figura 5.18).

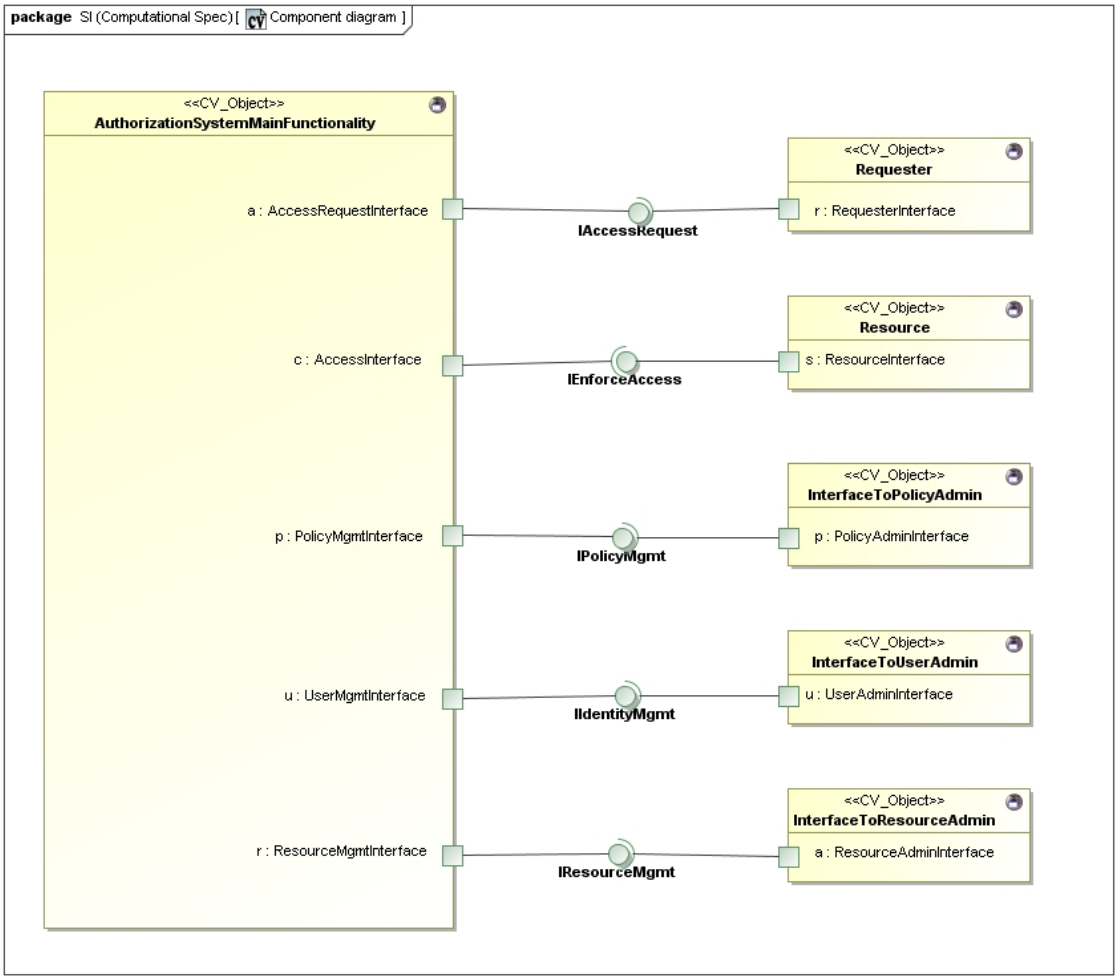

Figura 5.18. Objetos computacionales e interfaces del sistema de control de acceso

El objeto principal se ha denominado *Authorization System Main Functionality* y agrupa todos los elementos que forman parte de la infraestructura de control de acceso. Este objeto permite ver el mecanismo de control de acceso como un todo. Expone tres interfaces de gestión de información -una para políticas (*PolicyMgmtInterface*), otra para identidades de usuarios (*UserMgmtInterface*) y otra para información de recursos (*ResourceMgmtInterface*), una interfaz de petición de acceso (*AccessRequestInterface*) y otra de acceso a los recursos (*AccessInterface*). El resto de objetos son el

solicitante de acceso (*Requester*), el recurso (*Resource*) y los intermediarios de administración de políticas (*InterfaceToPolicyAdmin*), de información de usuarios (*InterfaceToUserAdmin*) y de información de recursos (*InterfaceToResourceAdmin*).

La Figura 5.19 muestra una vista interna del objeto *Authorization System Main Functionality.* Tanto la interfaz de petición de acceso como la de acceso al recurso son trasladadas al objeto computacional *Resource Access Manager.* Este objeto recibe las peticiones de los usuarios y devuelve las decisiones de acceso a los mismos. El gestor de acceso a recursos tiene además dos interfaces adicionales: una que utiliza para derivar la petición de acceso al *Context Manager* y recibir la decisión del sistema (*AccessDecisionRequestInterface*), y otra para comunicarse con el gestor de obligaciones (*Obligation Manager*) cuando la decisión de acceso está supeditada al cumplimiento de obligaciones (la interfaz es *ObligationInterface*). Las interfaces externas del objeto completo (es decir, aquellas dedicadas a la administración de políticas e información de usuarios y recursos) son delegadas a los objetos computacionales básicos que se especificaron siguiendo los requisitos de HISA que actuarán como proveedores de esas clases de información cuando sean necesarios. Así para las políticas tenemos el objeto computacional básico *Policy* (con una interfaz tanto para la gestión de las mismas como para su consulta), el objeto *Controlled element info* que hace lo mismo para información de recursos, y el objeto *User info* para la información de los usuarios. Además existe un objeto computacional, el *Environment Provider*, que proporciona información de contexto pero que no es administrada por los usuarios sino que se obtiene de forma automática del sistema y el entorno. Por su parte, el objeto computacional *Context Manager* recoge (y responde) las peticiones de acceso del *Resource Access Manager* a través de la interfaz *DecisionRequestInterface* y la información de los proveedores mediante la interfaz *InfoInterface.* La decisión de acceso será tomada por el objeto *Decision Agent* que recibe la información necesaria del *ContextManager* a través de la interfaz *DecisionInfoInterface* y las políticas del objeto *Policy* por la interfaz *PolicyRecoveryInterface.* La petición de acceso y la respuesta se pasan a través de la interfaz *DecisionInterface.*

Una vez descritos los objetos computacionales de este punto de vista es necesario especificar los métodos que operan en cada una de las interfaces. Las firmas de interfaces mediante las que se comunican los diversos objetos computacionales especifican interfaces de operación, existiendo tanto anuncios (*announcement*, donde el cliente no espera respuesta) como interrogaciones (*interrogation*, el servidor debe responder con un mensaje). Comentamos brevemente las operaciones que residen en cada firma de interfaz:

- *IAccessDecision*: la utiliza el gestor de acceso a recursos para pasarle la petición de acceso (tipo de dato *XACMLAuthzDecisionQuery*) al gestor de contexto. Recibe como respuesta un dato de tipo *XACMLAuthzDecisionStatement.*

- *IDecisionInfo*: el agente de decisión solicita información (*AttributeQuery*) al gestor de contexto y éste le responde con un dato tipo *AttributeStatement.*

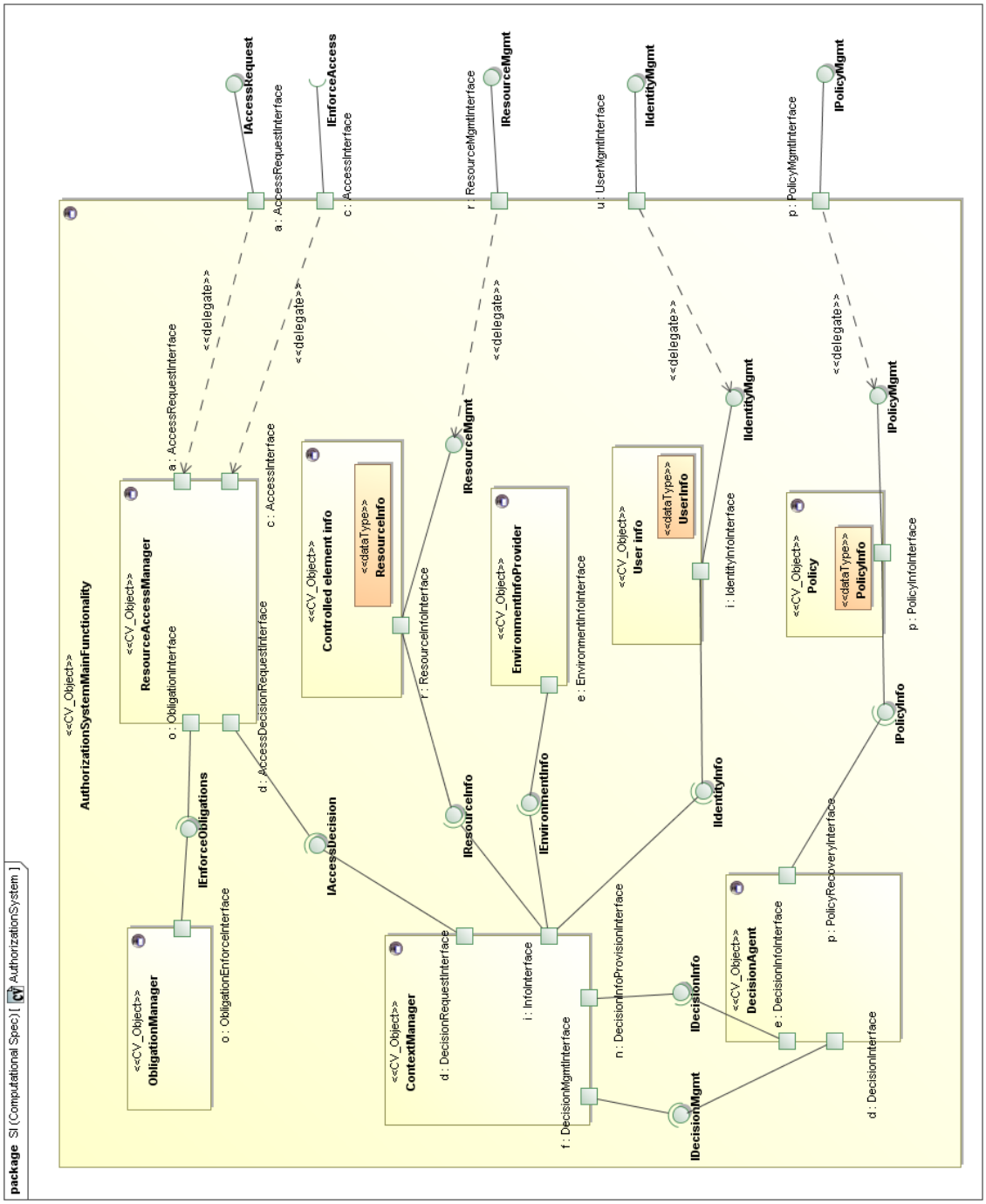

Figura 5.19. Vista interna del objeto computacional Authorization System Main Functionality

- *IDecisionMgmt*: el gestor de contexto pasa la petición de acceso (*XACMLAuthzDecisionQuery*) al agente de decisión. Una vez que éste realiza los procesos internos necesarios, responde con la decisión en formato *XACMLAuthzDecisionStatement.*

- *IEnvironmentInfo*: el gestor de contexto solicita información pasando un dato de tipo *AttributeQuery* al proveedor de información de contexto y la respuesta se devuelve en un *AttributeStatement.*

- *IAccessRequest*: un usuario solicita acceder a un recurso pasando su identificador (tipo *UserIdentifier*) al sistema de control de acceso. Una vez que se resuelve la petición, la decisión es devuelta al usuario como un dato de tipo *Decision.*

- *IEnforceObligations*: si en una decisión existen obligaciones, el gestor de acceso a recursos lo anuncia al gestor de obligaciones a través del tipo de dato *Obligation*.

- *IEnforceAccess*: tras una decisión positiva, el gestor de acceso al recurso anuncia a éste que una petición ha sido resuelta satisfactoriamente. En el mensaje se pasa un dato de tipo *AccessDecision.*

- *IIdentityInfo*/*IResourceInfo*: estas interfaces exponen, para las clases de información *User info* y *Controlled element*, los métodos básicos especificados por la norma HISA para la consulta de instancias de estas clases de objetos. Es decir, en cada interfaz se encontrarán los métodos *detail, Xdetail, Cdetail, BRdetail, Sdetail, list, Xlist, Clist, BRlist* y *Slist.* Los métodos aceptarán un dato de tipo *AttributeQuery* y devolverán como respuesta un *AttributeStatement.*

- *IIdentityMgmt*/*IResourceMgmt*: estas interfaces cubren las operaciones para la gestión de información de usuarios y recursos respectivamente. Heredan los métodos de las interfaces anteriores y añaden los correspondientes a la norma HISA de gestión (*add, Xadd, Cadd, BRadd, update, Xupdate, Cupdate, BRupdate, delete, Xdelete, Cdelete, BRdelete* y *pack*).

- *IPolicyInfo*: a través de un dato de tipo *XACMLPolicyQuery* el agente de decisión solicita las políticas aplicables a una petición de acceso para tomar una decisión. El objeto *Policy* las devuelve mediante un *XACMLPolicyStatement.* Esta interfaz expone los métodos *detail, Xdetail, Cdetail, BRdetail, Sdetail, list, Xlist, Clist, BRlist* y *Slist.*

- *IPolicyMgmt*: además de heredar el método de la firma de interfaz anterior, ésta recoge aquellas operaciones destinadas a gestionar las políticas. Consta de los métodos *add, Xadd, Cadd, BRadd, update, Xupdate, Cupdate, BRupdate, delete, Xdelete, Cdelete, BRdelete* y *pack*.

En las interfaces anteriores ya se han visto algunos de los tipos de datos que se intercambian entre objetos computacionales. Los tipos de datos heredados de la norma XACML tienen especial relevancia en este sistema de control de acceso como por ejemplo la solicitud de políticas de XACML (*XACMLPolicyQuery*), la sentencia de políticas de XACML (*XACMLPolicyStatement*) o el identificador de política (*PolicyIdentifier*). Para las decisiones se tienen la petición de decisión de XACML (*XACMLAuthzDecisionQuery*) y la sentencia de decisión XACML (*XACMLAuthzDecisionStatement*). Cuando se consulta un atributo mediante el objeto correspondiente (*AttributeQuery*) se puede pasar un identificador del recurso (*ResourceIdentifier*) o usuario (*UserIdentifier*) del cual se está pidiendo la información y la respuesta tiene el tipo sentencia de atributo (*AttributeStatement*).

4.2.5. Correspondencia entre los puntos de vista de la Empresa y Computacional

Las relaciones entre los objetos de empresa y los computacionales son tanto de elementos como de agentes del sistema:

- los agentes proveedores de información: *Policy Information System* con el objeto computacional *Policy*, *User Information System* con *User info, Resource Information System* con *Controlled element info,* y *Environment Information System* con *EnvironmentInfoProvider;*

- el elemento artefacto *Resource; y*

- el resto de agentes del sistema: *Policy Decision System* con *DecisionAgent, Context Handler System* con *ContextManager, Obligation System* con *ObligationManager,* y *Policy Enforcement System* con *ResourceAccessManager.*

4.2.6. Correspondencia entre los puntos de vista de la Información y Computacional

El último aspecto de la especificación del sistema de control de acceso que debemos abordar es el de las correspondencias entre los puntos de vista de la Información y Computacional. Estas relaciones son de *Access requester* con *Requester; Controlled element* con *Resource; Resource information system* con *Controlled element info; Policy decision system* con *DecisionAgent; Obligation system* con *ObligationManager; Environment information system* con *EnvironmentInfoProvider; User information system* con *User info; Context handler system* con *ContextManager; Policy information system* con *Policy;* y *Policy enforcement system* con *ResourceAccessManager.* También se relacionan los tipos de datos computacionales con clases de información como *Access control decision* con *AccessDecision*, *Policy* con *PolicyInfo*, *Controlled element info* con *ResourceInfo*, *Obligation policy* con *Obligation*, y *User info* con *UserInfo*.

Aparte de este grupo de correspondencias, deben especificarse las relaciones que estipula la norma HISA entre las clases de información y aquellos objetos computacionales que agrupan los métodos de gestión de los primeros. El objetivo de los *CorrespondenceLinks* es identificar el mismo componente del sistema en dos puntos de vista separados y se entiende que las relaciones entre las clases de información y los objetos computacionales gestores de instancias de esas clases no son objeto de este tipo de correspondencias.

5. Resultado 4 – La gestión semántica en HISA

5.1. Introducción

Una arquitectura abierta de servicios se compone de múltiples elementos dinámicos y heterogéneos, distribuidos geográficamente, que pertenecen a dominios administrativos separados y mantenidos por diferentes organizaciones. La interoperatividad entre tales elementos se debe basar, entre otras cosas,

en un entendimiento mutuo a todos los niveles (técnico, sintáctico, semántico e intencional). El establecimiento de un lenguaje común que pueda ser utilizado e interpretado por los sistemas de forma automática es uno de los requisitos esenciales para alcanzar dicha interoperatividad. La gestión de semántica es la capacidad de aportar significado a la información que los sistemas utilizan para desarrollar sus actividades o comunicarse entre sí. Esta información anotada permite mejorar la eficiencia de los procesos puesto que los sistemas son capaces de discriminar entre distintas soluciones, automatizar algunas de sus actividades o mejorar la cooperación con otros basándose en un mayor entendimiento de los requisitos y capacidades de los sistemas cooperantes. La gestión de semántica en una arquitectura abierta es una capacidad transversal a todos los servicios de la misma ya que puede dar soporte a una gran cantidad de procesos desde los de infraestructura de propósito general a los específicos del dominio de aplicación. En este apartado se especifica la incorporación de las capacidades de gestión de semántica a la arquitectura sanitaria establecida por la norma ISO12967. Los elementos que dan soporte a la gestión de semántica son formalizados de acuerdo con los principios de HISA, RM-ODP y UML4ODP. En el presente trabajo de Tesis Doctoral se exponen las tres vistas independientes de tecnología centradas en el sistema de gestión de semántica, ampliando los aspectos que recogía la norma con capacidades más generales y sofisticadas.

5.2. Punto de vista de la Empresa

La formalización de los elementos de soporte a la gestión de semántica podría partir de una nueva comunidad que forma parte de la agregación de la comunidad HIS como hemos especificado para los aspectos de autorización en el apartado anterior. Es lógico mantener todos estos elementos dentro de una comunidad puesto que su comportamiento e interacciones van orientados a un objetivo común: dar soporte a la gestión de semántica en arquitecturas de sistemas distribuidos y abiertos. Haciendo una revisión de la norma HISA se puede comprobar que, aun de forma limitada, estas capacidades están contempladas dentro del “grupo de actividades de clasificación, codificaciones y diccionarios”. Por lo tanto, el mecanismo más adecuado para incluir estas consideraciones es extender la comunidad ya existente en lugar de crear una nueva (Figura 5.20).

Los requisitos especificados en la vista de la empresa para las actividades relacionadas con la semántica del sistema indican que la arquitectura debe proveer de facilidades para los usuarios que cubran la gestión de:

- los distintos sistemas de clasificación y tipos utilizados para los conceptos del sistema de información,

- los vocabularios comunes que cubren el conjunto de términos que las aplicaciones necesitan y utilizan para describir el dominio de aplicación,

- las dependencias y reglas que pueden existir entre conceptos relacionados así como entre instancias o individuos en función de los valores concretos de sus atributos,

- las reglas que indican cómo instanciar los conceptos del dominio, y

- la correspondencia entre elementos del modelo de información de la arquitectura y los mensajes y vistas requeridos para soportar interacciones con otros sistemas o entre componentes de distintos dominios de la organización. Normalmente estos mensajes y vistas se basan en estándares de comunicación y modelos de datos normalizados.

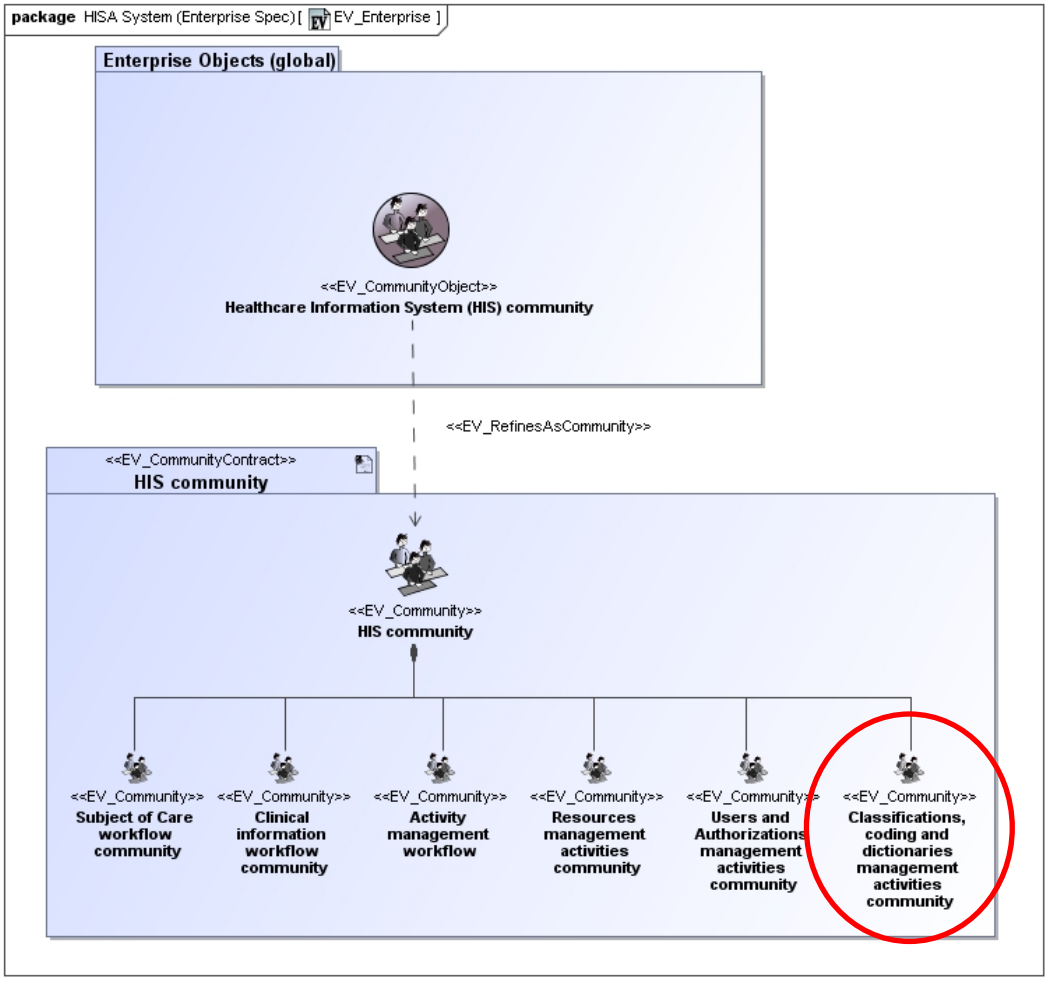

Figura 5.20. Comunidad de capacidades semánticas formando parte de la comunidad HIS

Como consecuencia se identifican dos conjuntos de requisitos:

1. Para cada flujo de trabajo y grupo de actividades de usuario, se deben proveer de capacidades que permitan la gestión de diccionarios, clasificaciones y reglas adoptadas por la organización para los procesos implicados.

2. De forma global deben procurarse herramientas y mecanismos para facilitar la gestión de dependencias y relaciones entre conceptos con propósitos clínicos, organizativos y de gestión, también cubriendo múltiples áreas y el mapeo con otros estándares específicos.

La norma HISA plantea un modelo en el que cada componente especificado es modular, consistente y autosuficiente, proporcionando las capacidades necesarias para alguna de las actividades de usuario que se identifican en el punto de vista de la Empresa. Se pretende así reducir al máximo las dependencias entre componentes y permitir que puedan ser incorporados de forma aislada a distintas arquitecturas sin pérdida de capacidades. Este modelo, por tanto, exige que cada componente incorpore capacidades de gestión de semántica para las actividades de usuario concretas en las que participe evitando la dependencia con otros agentes externos y aumentando la modularidad de la arquitectura. La fuerte transversalidad de las actividades de gestión semántica (aparecen en todos y cada uno de los componentes de la arquitectura) provoca que el desarrollo de componentes para estas actividades sea reutilizado en las demás, de manera que un desacoplo total de las mismas es prácticamente imposible. Para potenciar la interoperatividad semántica en escenarios multi-dominio hacemos uso del concepto de federación. En este esquema los elementos en cada dominio (los cuales pertenecen a cada componente de la arquitectura) se constituyen como agentes federados. Para mantener la coherencia entre los mismos se establecen correspondencias entre los artefactos semánticos que utilizan en sus dominios y los de la federación. Este modelo puede verse en la Figura 5.21 y la relación entre los elementos federados se desarrollará más adelante. La gestión semántica es especialmente necesaria en una arquitectura que pretenda seguir el modelo federado ya que uno de los mayores problemas a resolver es la reconciliación semántica de los sistemas que se van a integrar, que debe hacerse además sin modificar ninguno de ellos. Es pues imprescindible que en la arquitectura existan componentes que ayuden a solucionar estos conflictos.

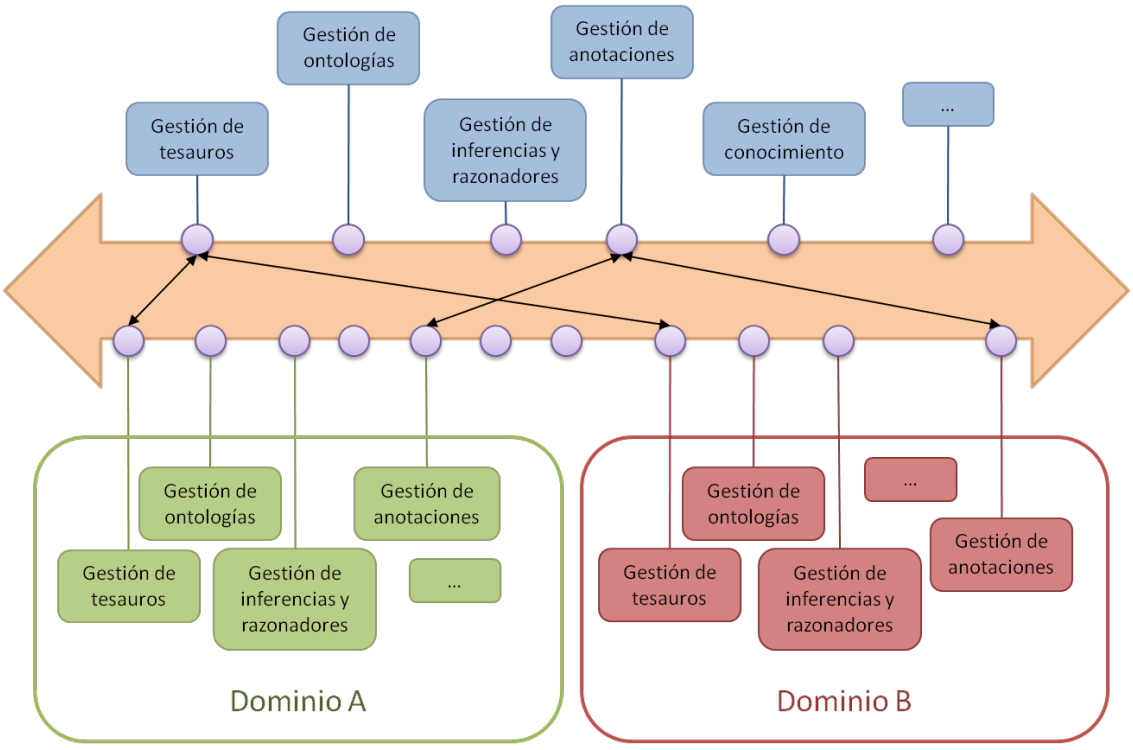

Figura 5.21. Principales capacidades semánticas federadas (arriba) y autónomas o locales (abajo)

En el presente trabajo de Tesis Doctoral y siguiendo la propuesta de [189] se amplían los requisitos de actividades semánticas recogidos en el punto de vista de la Empresa de la ISO12967 al considerar que los avances realizados en cuanto a gestión semántica deben ser tenidos en cuenta al desarrollar un estándar de esta envergadura. Ante todo y a partir de ahora, además de referirnos a sistemas de clasificación, códigos o diccionarios, como la norma indica, hablaremos también de la gestión de ontologías. La ventaja es que el concepto de ontología, definido como el conjunto de conceptos y relaciones entre ellos dentro de un dominio, tiene mayor capacidad de expresar semántica. Además puede incluir el conjunto de individuos (o instancias) de ese dominio. Algunas de las principales funciones de gestión de semántica útiles en la arquitectura son:

- **Gestión de tesauros:** se incluyen en este grupo todos los componentes tradicionales de gestión de sistemas de términos y sistemas de codificación.

- **Gestión de ontologías:** estos componentes gestionan repositorios de ontologías permitiendo a los usuarios, por ejemplo, añadir, consultar o eliminar conceptos y relaciones de las ontologías. Pueden asimismo proveer facilidades para la correspondencia o fusión de distintas ontologías, pieza clave para la sostenibilidad del modelo federado en el que los componentes autónomos utilizarán ontologías locales a sus dominios y deberán relacionarlas con la ontología común de la federación. Las tareas de correspondencia y fusión pueden ser manuales, automáticas o semiautomáticas.

- **Gestión de inferencias y razonadores:** serían agentes del sistema que ayudan a definir reglas de inferencia y a ejecutarlas cuando es preciso. En muchas ocasiones son la pieza fundamental para el comportamiento proactivo de los componentes de la arquitectura, ya que permiten que, al observar algún cambio en las condiciones del entorno, los componentes puedan de forma autónoma ejecutar alguna acción concreta.

- **Gestión de anotaciones:** los componentes que proveen la capacidad de gestión de anotaciones permiten añadir aspectos semánticos (meta-información) a las piezas de información almacenada en el sistema para su gestión. Las anotaciones son la base para la automatización de un conjunto de funciones de gestión de semántica e inferencia de nueva información.

- **Gestión de conocimiento:** la información anotada semánticamente, y por tanto procesable y entendible por los sistemas, entra en la categoría de conocimiento. Existirán componentes encargados de realizar las tareas de gestión de ese conocimiento (creación, modificación y borrado) así como en ponerlo en uso para la mejora en la eficiencia y automatización de procesos tanto de gestión como de análisis, planificación, etc.

La gestión de semántica tiene una marcada transversalidad y todas estas funciones pueden influir (e influyen) en el resto de componentes de la arquitectura. Así por ejemplo, con el crecimiento de la

arquitectura se necesitan capacidades para la creación de nuevos componentes y aplicaciones a partir de los disponibles y, por supuesto, para la publicación y provisión de las nuevas capacidades disponibles. Las tareas de composición, descubrimiento e invocación de componentes pueden ser realizadas de forma agnóstica a la semántica pero, cuanto más crezca la carga semántica en la especificación de interfaces y componentes y más incorporado esté su uso en esas funciones, más eficientes y automáticas podrán llegar a ser.

5.2.1. Formalización del punto de vista de la Empresa de la gestión de semántica

La forma de abordar la especificación de la comunidad para la gestión de semántica en la arquitectura de referencia va a ser extender el grupo de actividades de codificaciones, clasificaciones y diccionarios. Este grupo ha sido modelado como una comunidad que forma parte de la agregación que es la comunidad HIS. En primer lugar se ha modificado el nombre de la subcomunidad a *Semantic Management Community* para que acomode la noción de gestión semántica la cual va más allá que la de gestión de diccionarios, codificaciones y clasificaciones. En las tareas de gestión de clasificaciones, criterios de codificación y diccionarios, la norma HISA incluye, casi en exclusiva, la gestión de sistemas de tesauros obviando otros elementos semánticos como la gestión de ontologías, la anotación de la información, la organización de modelos de información, etc. Por ello se refina la comunidad de gestión de semántica ampliando así el concepto para que cubra no sólo tesauros sino también otros elementos semánticos más complejos como ontologías.

El objetivo de la comunidad es el de "dar soporte a la gestión de cualquier elemento de semántica (ontologías, anotaciones, modelos de información, etc.) de forma transversal a todos los servicios y sistemas de la arquitectura". A alto nivel esta comunidad va a favorecer la interoperatividad semántica entre componentes dispares de la arquitectura y, a priori, esto se alcanza a través de una de las dos propuestas siguientes:

1. La estandarización de los modelos de información forzando la homogeneidad de la semántica; esto es, el establecimiento de un modelo de información único que cubre todos los aspectos de la organización el cual deben utilizar todos los elementos sin modificaciones ni extensiones. Esta propuesta reduce la complejidad de los procesos pero limita la evolución y apertura de la arquitectura a nuevas soluciones y elementos.

2. La mediación entre modelos de información diferentes en base a traducciones o transformaciones. Esta es la propuesta del modelo federado en el cual los elementos utilizan un modelo de información local a su dominio que debe ser traducido o fusionado con el modelo de información federado para permitir la comunicación con otros elementos de la arquitectura. Esta propuesta exige la incorporación de capacidades para la traducción o fusión de modelos de información pero potencia la autonomía de los distintos elementos y facilita la incorporación de nuevos componentes que no tendrán que ser modificados para colaborar dentro de la arquitectura.

Una combinación de ambas propuestas es posible y es la solución que se contempla en lo sucesivo: se estandarizan los conceptos básicos independientes del dominio o contexto de negocio en el sistema de información federado, y para aquellos conceptos dependientes de dominio y que difieren entre distintas aplicaciones se utilizan interceptores entre sistemas de información siguiendo el modelo federado. Los procesos identificados a partir de las funciones de gestión de semántica se recogen en la Figura 5.22 y se pueden agrupar en:

- Gestión de ontologías: comprenden la creación, modificación, evaluación, recuperación y borrado de ontologías. Se incluyen en este grupo aquellos procesos que soportan el modelo federado como el mapeo o traducción entre ontologías.

- Gestión de tesauros: comprendiendo el manejo de sistemas de términos y codificaciones tradicionales y heredando los aspectos especificados en la norma ISO12967.

- Gestión de anotaciones: creación y modificación de anotaciones sobre elementos de información.

- Gestión de inferencias: para el manejo y ejecución de reglas de inferencia.

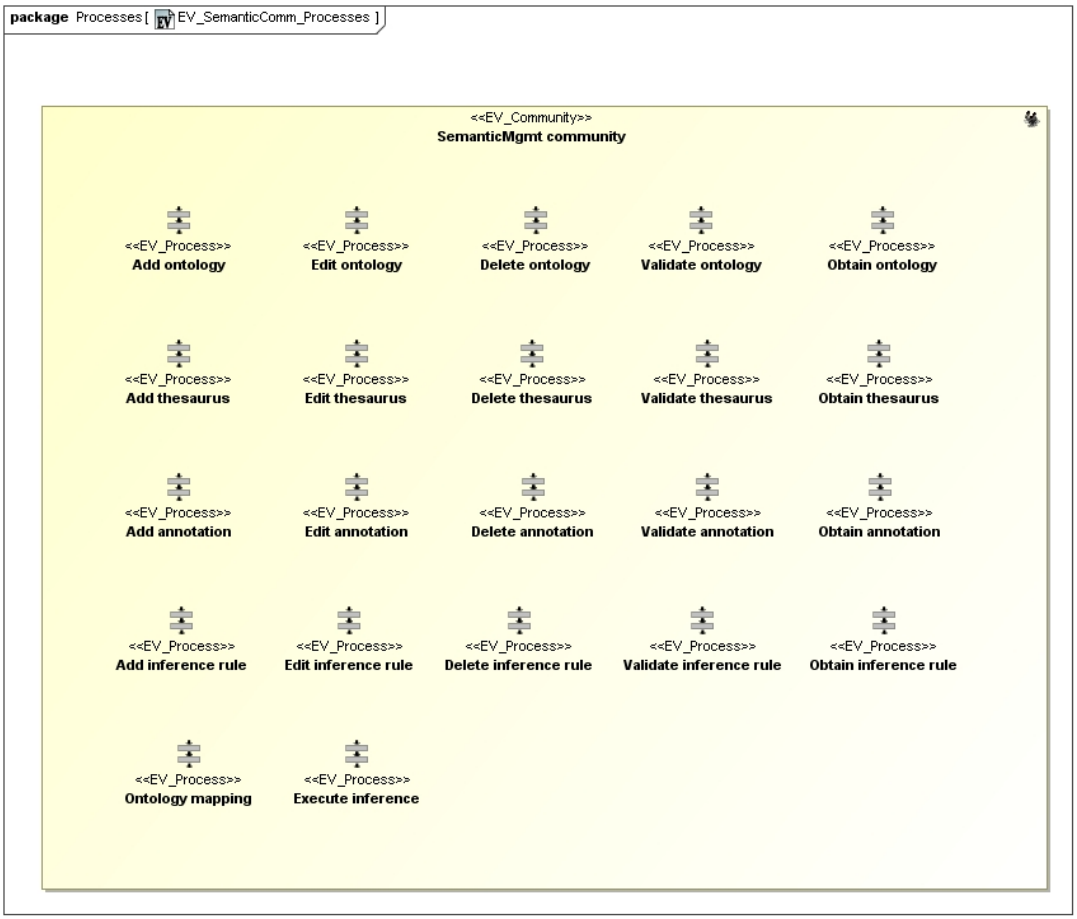

Figura 5.22. Punto de vista de la Empresa – procesos de la comunidad de gestión de semántica

Las funciones de gestión de conocimiento y otras generales que se benefician de la introducción de semántica (como las de composición y descubrimiento de componentes) quedan fuera de esta especificación ya que se basan en las capacidades de gestión de semántica otorgadas por las anteriores para ofrecer capacidades avanzadas y complejas.

A continuación se detallan cada uno de estos procesos, especificando qué roles de comportamiento están involucrados en los mismos y qué artefactos se manejan.

1. Añadir ontología (*Add ontology*): a través de este proceso se incluye una nueva ontología en el sistema. Este proceso es iniciado por un actor con el rol de gestor de ontologías (*Ontology manager*) el cual se pone en contacto con el elemento que almacena y mantiene las mismas (con rol proveedor de ontologías, *Ontology provider*). El primero pasa un artefacto con la ontología y solicita su inclusión en el sistema. Se realiza el proceso interno de inclusión y el proveedor devuelve una respuesta con el resultado del proceso. Si éste ha sido exitoso el administrador recupera el identificador de la ontología en el sistema y termina el proceso. En otro caso se especificará en la respuesta la razón del rechazo.

2. Editar ontología (*Edit ontology*): este proceso permite modificar una ontología ya incluida en el sistema. El gestor de ontologías (*Ontology manager*) comunica al proveedor (*Ontology provider*) su deseo de editar una ontología. Este último realiza el proceso interno necesario y responde al solicitante con el resultado de la modificación.

3. Eliminar ontología (*Delete ontology*): este proceso de empresa permite eliminar una ontología del sistema. Ahora el gestor solicita la eliminación de la ontología de la cual pasa su identificador al proveedor. Éste realiza el proceso apropiado y devuelve la respuesta indicando el éxito o fracaso de la petición.

4. Validar ontología (*Validate ontology*): una vez que una ontología está aceptada en el sistema es necesario que sea validada para poder utilizarla. En este caso tanto el gestor de ontologías como cualquier componente con rol de usuario (*User*) puede solicitar al proveedor de ontologías que realice el proceso de validación. Para ello se le pasa el identificador de la ontología en el sistema, recibiendo como respuesta el resultado de este proceso. Si la ontología ha quedado validada entonces puede utilizarse en el sistema. En caso contrario queda invalidada y es imposible su uso.

5. Obtener ontología (*Obtain ontology*): cualquier elemento de la arquitectura con el rol de usuario o el gestor de ontologías puede consultar una ontología del sistema. A través de este proceso un componente indica el identificador de la ontología que quiere recuperar al proveedor y éste la devuelve si no existe ningún error.

6. Mapeo de ontologías (*Ontology mapping*): este proceso es esencial en el modelo federado pues permite realizar traducciones entre conceptos de las ontologías. Cualquier componente con el rol

de usuario o de gestor de ontologías puede solicitar un mapeo entre conceptos de ontologías indicándole al proveedor la ontología local a corresponder con la federada.

7. Añadir tesauro (*Add thesaurus*): a través de este proceso se incluye un nuevo tesauro en el sistema. Este proceso es iniciado por un actor con el rol de gestor de tesauros (*Thesaurus manager*) el cual se pone en contacto con el elemento que los almacena y mantiene y cuyo rol es el de proveedor de tesauros (*Thesaurus provider*). El primero pasa un artefacto con el tesauro y solicita su inclusión en el sistema. Se realiza el proceso interno de inclusión y el proveedor devuelve una respuesta con el resultado del proceso. Si éste ha sido exitoso el administrador recupera el identificador del tesauro en el sistema y termina el proceso. En otro caso se especificará en la respuesta la razón del rechazo.

8. Editar tesauro (*Edit thesaurus*): este proceso permite modificar un tesauro ya incluido en el sistema. Para ello, el gestor comunica al proveedor su deseo de editar un tesauro y éste realiza el proceso interno necesario y responde al solicitante con el resultado de la modificación.

9. Eliminar tesauro (*Delete thesaurus*): este proceso de empresa permite eliminar un tesauro del sistema. Ahora el gestor pasa al proveedor el identificador del elemento del cual solicita la eliminación. El proveedor realiza el proceso apropiado y devuelve la respuesta indicando el éxito o fracaso de la petición.

10. Validar tesauro (*Validate thesaurus*): al igual que ocurre con las ontologías, para que un tesauro pueda ser utilizado en el sistema debe ser previamente validado. Tanto el gestor de tesauros como cualquier componente con rol de usuario puede solicitar al proveedor correspondiente que realice el proceso de validación. Para ello se le pasa el identificador del tesauro, recibiendo como respuesta el resultado de este proceso. Si el tesauro ha quedado validado entonces puede utilizarse en el sistema. En caso contrario queda invalidado y es imposible su uso.

11. Obtener tesauro (*Obtain thesaurus*): cualquier elemento de la arquitectura con el rol de usuario o el gestor de tesauros puede consultar un tesauro del sistema. A través de este proceso un componente indica el identificador del mismo y el proveedor se lo devuelve si no existe ningún error.

12. Añadir anotación (*Add annotation*): mediante este proceso, un componente de la arquitectura con el rol de comportamiento de gestor de anotaciones propone al proveedor de anotaciones la inclusión de una nueva anotación en el sistema. Cuando el proveedor verifica el nuevo elemento devuelve el resultado de la inclusión al gestor, ya sea indicándole que la anotación fue aceptada (*accepted*) o rechazada (*refused*).

13. Editar anotación (*Edit annotation*): el gestor de anotaciones utiliza este proceso para modificar alguna anotación que ya está incluida y aceptada en el sistema. Para ello comunica al proveedor de anotaciones cuáles son los cambios que debe realizar y éste le devuelve el resultado de la modificación.

14. Eliminar anotación (*Delete annotation*): si en lugar de querer modificar una anotación el gestor desea eliminar una del sistema sólo tiene que comunicarlo al proveedor de anotaciones mediante este proceso.

15. Validar anotación (*Validate annotation*): que una anotación haya sido aceptada en el sistema por el proveedor de anotaciones no significa que pueda ser utilizada. Previamente debe ser validada a través de este proceso que puede ser iniciado tanto por el gestor de anotaciones o el propio proveedor como por cualquier componente con rol de usuario. El proveedor utiliza el identificador de la anotación para discriminarla y manda como respuesta el resultado del proceso de validación. Si la anotación ha quedado validada entonces puede utilizarse en el sistema. En caso contrario queda invalidada y es imposible su uso.

16. Obtener anotación (*Obtain annotation*): cualquier componente de la arquitectura con rol de usuario o el gestor de anotaciones puede solicitar una anotación al proveedor de las mismas. Para ello el componente pasa al proveedor cierta información para identificarla y éste devuelve la anotación o un error si no existe tal elemento en el sistema.

17. Añadir regla de inferencia (*Add inference rule*): análogamente a como se realiza la inclusión de nuevos elementos semánticos en el sistema, a través de este proceso un componente con rol de gestor de inferencias solicita al proveedor la inclusión de una nueva regla de inferencia en el sistema. Se realiza entonces un proceso interno de verificación de la regla de inferencia en el que se evalúa su formato. Este proceso es distinto del de validación (ver proceso número 20) el cual está sujeto a condiciones de aplicabilidad de la regla de inferencia con respecto a las ontologías en uso en el sistema. Una vez que finaliza el proceso de verificación, el proveedor responde al gestor indicando si la regla fue aceptada (*accepted*) o rechazada (*refused*).

18. Editar regla de inferencia (*Edit inference rule*): este es el proceso por el cual el gestor de inferencias solicita la modificación de una regla de inferencia ya aceptada en el sistema. El proveedor de inferencias responde con el resultado de la edición.

19. Eliminar regla de inferencia (*Delete inference rule*): para eliminar una regla de inferencia del sistema el gestor de inferencias utiliza este proceso en el que identifica la regla a eliminar y es el proveedor el que lleva a cabo el proceso de borrado.

20. Validar regla de inferencia (*Validate inference rule*): al incluir una regla de inferencia en el sistema (proceso número 17) se evalúa el formato de la regla y si está construida adecuadamente. Si la regla de inferencia es aceptada en el sistema el siguiente paso es validarla para que pueda ser utilizada. Este proceso de validación es iniciado por un usuario, el gestor de inferencias o el propio proveedor, y tiene en cuenta las condiciones de aplicabilidad establecidas en el sistema para las reglas de inferencia (por ejemplo, que la regla esté correctamente relacionada con ontologías en uso o que haya sido creada por una entidad autorizada). Una vez que el proveedor termina la validación devuelve a quien solicitó el proceso el resultado del mismo. Si la validación fue exitosa entonces la regla de inferencia pasa al estado validada (*validated*). En caso contrario queda invalidada (*disqualified*).

21. Obtener regla de inferencia (*Obtain inference rule*): cualquier componente de la arquitectura con rol de usuario o el gestor de inferencias puede solicitar una regla de inferencia al proveedor de las mismas indicándole cuál desea recuperar. Si existe el proveedor se la devuelve al componente que la solicitó. Si no se responde con un error.

22. Ejecutar inferencia (*Execute inference*): el último proceso de empresa es el de ejecución de inferencias. Es iniciado por un usuario o el gestor de inferencias los cuales solicitan al proveedor que ejecute el motor de inferencias. El proveedor recupera la/s ontología/s necesaria/s, las reglas de inferencia aplicables a dichas ontologías y ejecuta la inferencia. Devuelve el resultado del proceso (nueva información inferida) al elemento que inició el proceso.

A partir de los procesos descritos podemos identificar un conjunto básico de roles de comportamiento que se deben dar dentro de la comunidad de gestión de semántica. En concreto existe un proveedor (*provider*) y un gestor (*manager*) para cada artefacto de la comunidad, esto es, para las ontologías, tesauros, anotaciones y reglas de inferencia. Se tiene además un rol general de usuario (*User*) y otro para la ejecución de inferencias (*Inference engine*). En cuanto a los objetos de empresa pertenecientes a esta comunidad, por un lado están los actores entre los que se encuentran los puntos de información y administración de los diferentes tipos de elementos semánticos así como los usuarios individuales. Los objetos de empresa que representan artefactos también están recogidos y se han añadido algunos nuevos:

- Perfil semántico (*Semantic profile*): se corresponde con las anotaciones semánticas a elementos de información, recursos, individuos, etc.

- Arquetipo (*Archetype*): elemento semántico que facilita el uso de ontologías para expertos del dominio y que está siendo muy utilizado en el entorno de los sistemas de información sanitaria.

- Modelo de referencia (*Reference model*) y Modelo de dominio (*Domain model*): representan las dos caras de los modelos de información en el paradigma federado, es decir, el modelo de

referencia es el general de la arquitectura mientras que el modelo de dominio es el local que posee cada componente de la misma. La traducción entre ambos se realiza a través de las Reglas de mapeo (*Mapping rules*).

Los cuatro objetos de empresa que representan elementos semánticos (*Ontology*, *Thesaurus*, *Semantic Profile* e *Inference Rule*) presentan un ciclo de vida dividido en fases que indican diferentes estados de aplicación. En una primera aproximación el conjunto de estados es el mismo para todos los objetos de empresa: primero se propone la inclusión de la instancia concreta del objeto en el sistema (*proposed for inclusion*); una vez verificado mediante los procesos pertinentes se acepta (*accepted*) o rechaza (*refused*); aun aceptado debe ser a continuación validado para poder ser utilizado en el sistema (*proposed for validation*); si la validación falla entonces la instancia es invalidada (*disqualified*), de otro modo puede ser utilizada (*validated*); finalmente el ciclo de vida natural termina cuando la instancia es eliminada (*deleted*).

Finalmente no se ha considerado necesaria la introducción de políticas de empresa que restrinjan los comportamientos de los actores o el uso de los objetos de empresa.

5.3. Punto de vista de la Información

5.3.1. Aspectos heredados y modificados del punto de vista de la Información de HISA

HISA especifica que dentro de la arquitectura se debe integrar y organizar la información compartida por los agentes en un modelo de información consistente que soporte todas las necesidades de las actividades identificadas en el punto de vista de la Empresa. El modelo de información es entonces la base para lograr la integridad semántica, es decir, los conceptos y relaciones que se presentan en el punto de vista de la Información tienen que ser entendidos por los componentes de la arquitectura sin posibilidad de confusión. De acuerdo con el paradigma de federación existirá un modelo de información local para cada componente al margen del modelo de información de la arquitectura. Será necesario entonces (como se ha especificado en el punto de vista de la Empresa) que existan mecanismos que permitan establecer correspondencias desde los modelos locales al federado. Éste último, normalizado en HISA, sólo se extiende cuando aparece un concepto completamente nuevo, no recogido previamente, lo que evita redundancias. Cada componente de la arquitectura está obligado a utilizar este modelo de información común de manera que se incorporan los mecanismos necesarios para realizar transformaciones al modelo de información federado. Si además se publica el modelo federado como una ontología se facilita que los componentes locales puedan comprender el modelo federado y realizar estas transformaciones.

La arquitectura debe ser capaz de gestionar la información manejada diariamente en las actividades de usuario pero también la base de conocimiento que permite manejar la semántica (conceptos, relaciones, vocabularios y reglas) necesaria para instanciar un elemento concreto de la información

operacional requerida en la organización. Para cubrir estas necesidades, el punto de vista de la Información de HISA especifica que para cada modelo se pueden distinguir:

- **Objetos operacionales:** modelan las entidades implicadas en las actividades diarias dentro de la organización tanto para el tratamiento de sujetos de asistencia como para el correcto funcionamiento del sistema.

- **Objetos descriptivos:** especifican la semántica de los objetos operacionales. Modelan las entidades requeridas para el mantenimiento de la base de conocimiento en la que la organización se apoya para ejecutar sus actividades diarias tanto para la asistencia a los individuos como para la organización del funcionamiento interno.

En la Figura 5.23 se muestra la relación entre los dos tipos de objetos donde un objeto operacional es clasificado por uno descriptivo. Esta relación se representa en HISA con una dependencia que indica que el objeto operacional es una instancia del descriptivo. Además un objeto operacional es descrito por uno o varios objetos descriptivos e incluso pueden existir relaciones semánticas entre objetos descriptivos. Esta forma de modelar la semántica definida en el punto de vista de la Información es el mecanismo que utiliza HISA para la planificación de la gestión de semántica. En la extensión de los aspectos de gestión de semántica presentados en esta Tesis Doctoral se propone un paradigma de diseño que proporciona una mayor capacidad de expresión e interpretación semántica sobre la base de los objetos operacionales de HISA.

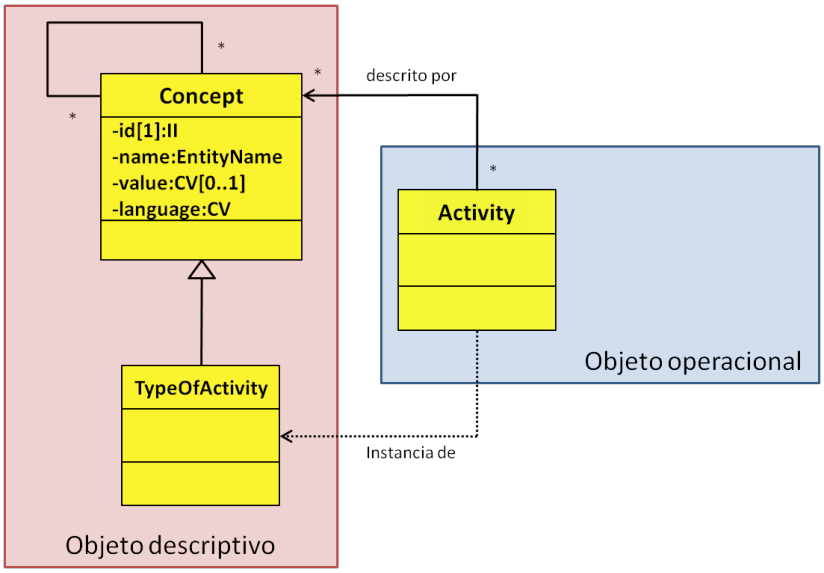

Figura 5.23. Relación entre objetos descriptivos y operacionales

El modelo de información propuesto por HISA como base de la gestión semántica es muy simple. Según lo que estipula la norma se debe contar con una base de conocimiento compuesta de objetos de información descriptivos ("*type of XXXX*") que equivalen a meta-objetos. Incluyen clasificaciones generales de conceptos clínicos, reglas que rigen la realización de actividades y otros. Los objetos de

información descriptiva modelan las entidades requeridas para la base de conocimiento. Todos los objetos descriptivos heredan de la clase principal *Concepto* que tiene como argumentos un nombre, un identificador, el idioma codificado y el valor codificado. Para cada objeto de información operacional se prevé un objeto de información descriptivo (el primero es una instancia del segundo, Figura 5.24) que contiene los principales datos de clasificación, propiedades, reglas y valores por defecto necesarios para la gestión de datos reales instanciados en el objeto operacional. Sobre estos objetos operacionales se pueden establecer clasificaciones y relaciones que es lo que permite añadir semántica. Además cada clase y atributo puede necesitar ser clasificado de acuerdo a diferentes y/o múltiples clasificaciones multi-lenguaje para diferentes propósitos (epidemiología, clínico, estadístico, etc.). Para soportar este requisito el modelo HISA proporciona el paquete *Conceptual Information Objects* capaz de organizar múltiples clasificaciones, terminologías y otros conceptos.

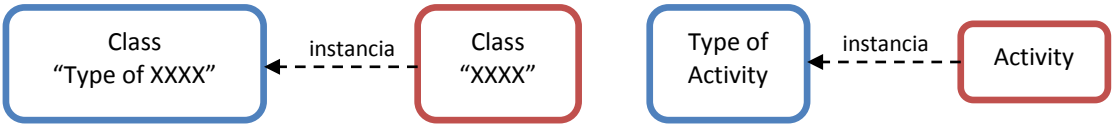

Figura 5.24. Relación entre objetos descriptivos y operacionales

El diagrama integrado que plantea HISA puede verse en la Figura 5.25. El elemento Criterio de clasificación (*Classification criteria*) permite clasificar las clases y/o atributos individuales de acuerdo a múltiples criterios de clasificación definidos en la clase Concepto (*Concept*) del modelo. El elemento Conjunto de atributos estructurados (*Structured attributes set*) identifica el atributo específico de la instancia que es clasificado con los criterios de clasificación. Si no se especifica ningún atributo, la clasificación se aplica a toda la instancia de la clase. Por último, Concepto especifica el elemento de clasificación relacionado con la instancia y el atributo (si se ha especificado uno). La clasificación propiamente dicha está implícita en las relaciones entre conceptos.

5.3.2. Formalización del punto de vista de la Información para la gestión de semántica

La gestión de semántica es una actividad muy transversal, es decir, que se realiza en realidad en todos los grupos de actividades. Por eso el estándar HISA comienza definiendo las clases descriptivas y operacionales fuera de cualquier modelo de información (Figura 5.25), porque este mecanismo de expresar semántica debe usarse en todos los modelos. En el presente trabajo de Tesis Doctoral, y siguiendo anteriores esfuerzos [189], se propone un cambio de paradigma de trabajo para definir los modelos de información. Este cambio afecta transversalmente a todos los modelos de información descritos hasta ahora y los sucesivos.

El objetivo del punto de vista de la Información debe ser proporcionar una ontología flexible y extensible que incluya los conceptos y relaciones principales del dominio. El beneficio de diseñar una ontología en lugar de un modelo de información tradicional es que se facilita la gestión y extensión mientras se aumenta la capacidad de expresión semántica. La ontología define qué conceptos, o clases, son

comunes a todos los componentes y qué relaciones existen entre estos conceptos. El modelo de información sólo define las clases; los objetos concretos serán instancias de las mismas y no forman parte del modelo de información. Del mismo modo en la ontología está clara la separación entre instancias y clases. Sin embargo, muchas herramientas de gestión actuales permiten manejar también las instancias, lo que aporta la ventaja de poder usar de forma integrada razonadores que faciliten la gestión del conocimiento. Éstos componentes pueden llegar a conclusiones a partir del análisis de las instancias y hechos existentes en el sistema y actuar en consecuencia, añadiendo nuevos hechos, instancias o ejecutando alguna otra acción.

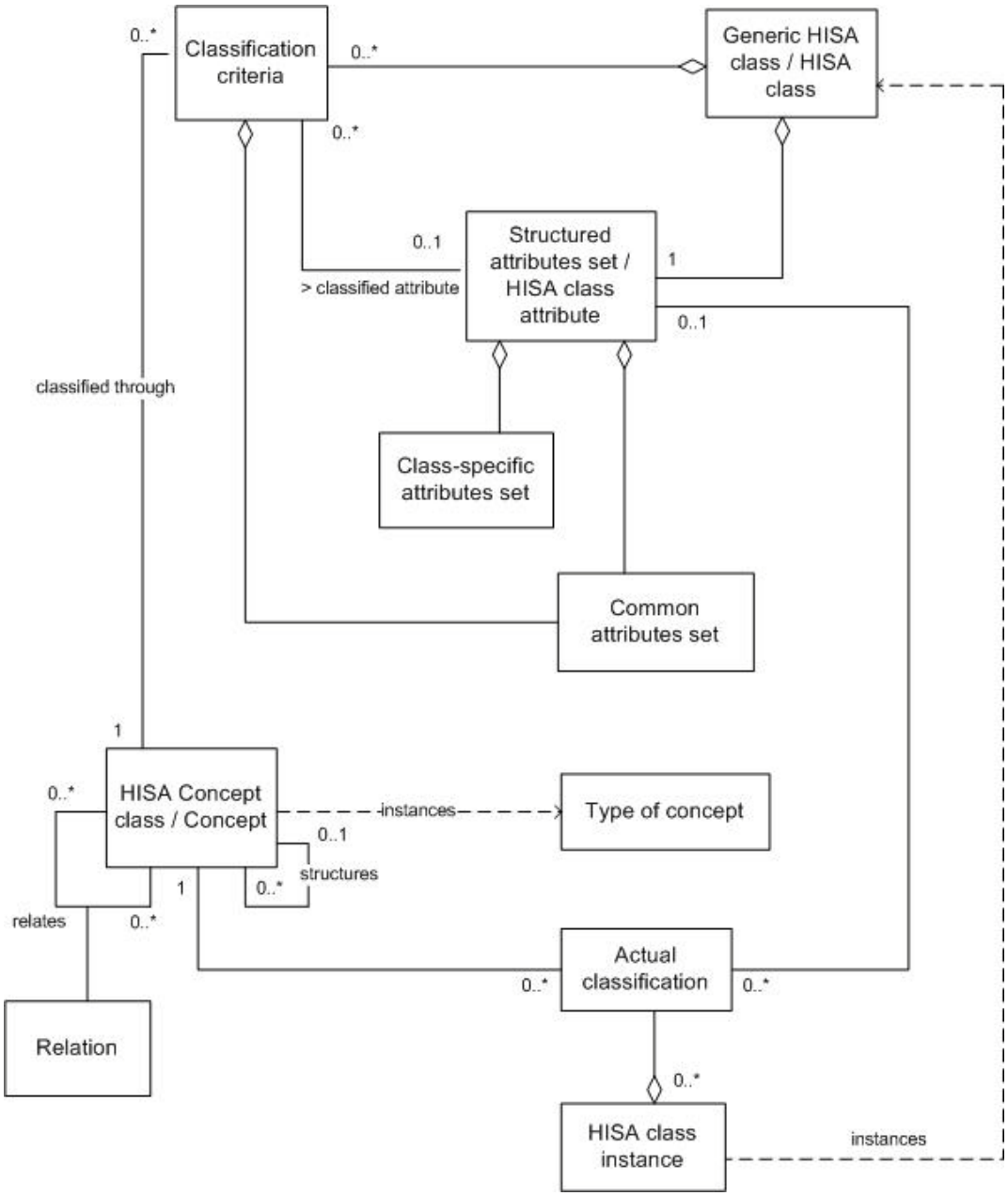

Figura 5.25. Diagrama integrado de HISA para la gestión de semántica

Si el modelo de información se define como una ontología, automáticamente todas las clases se convierten en conceptos y no es necesario utilizar la clase descriptiva *Concepto* ni las clases hijas *Tipo de XXXX*. Considerando que la capacidad de gestión semántica se proporciona en HISA en aquellas clases que se consideran instancias de alguna clase descriptiva *Tipo de XXXX* la ventaja de considerar absolutamente todas las clases como conceptos es evidente: todas las clases pueden ser gestionadas semánticamente de manera que la flexibilidad y el dinamismo que se buscaba en el modelo de información están asegurados.

El modelo de información semántico definía las relaciones entre las clases *Tipo de XXXX* pero esto ya no es necesario ya que las relaciones se establecen directamente entre cualesquiera conceptos del dominio a través de propiedades. ¿Qué debe incluir entonces el modelo de información semántico? Pues aquellos conceptos que se utilicen sólo para las tareas de gestión semántica y que no pertenezcan a ningún grupo de actividades. En la Figura 5.26 se representa un modelo resumido de los conceptos genéricos para cualquier tipo de ontología. Dado que cualquier clase definida formalmente en HISA será subclase de la clase *GenericHISAClass* (también denominada *HISAClass*), y por tanto será una clase de la ontología federada, será mucho más sencillo incluir nuevas clases y propiedades así como definir relaciones entre ellas.

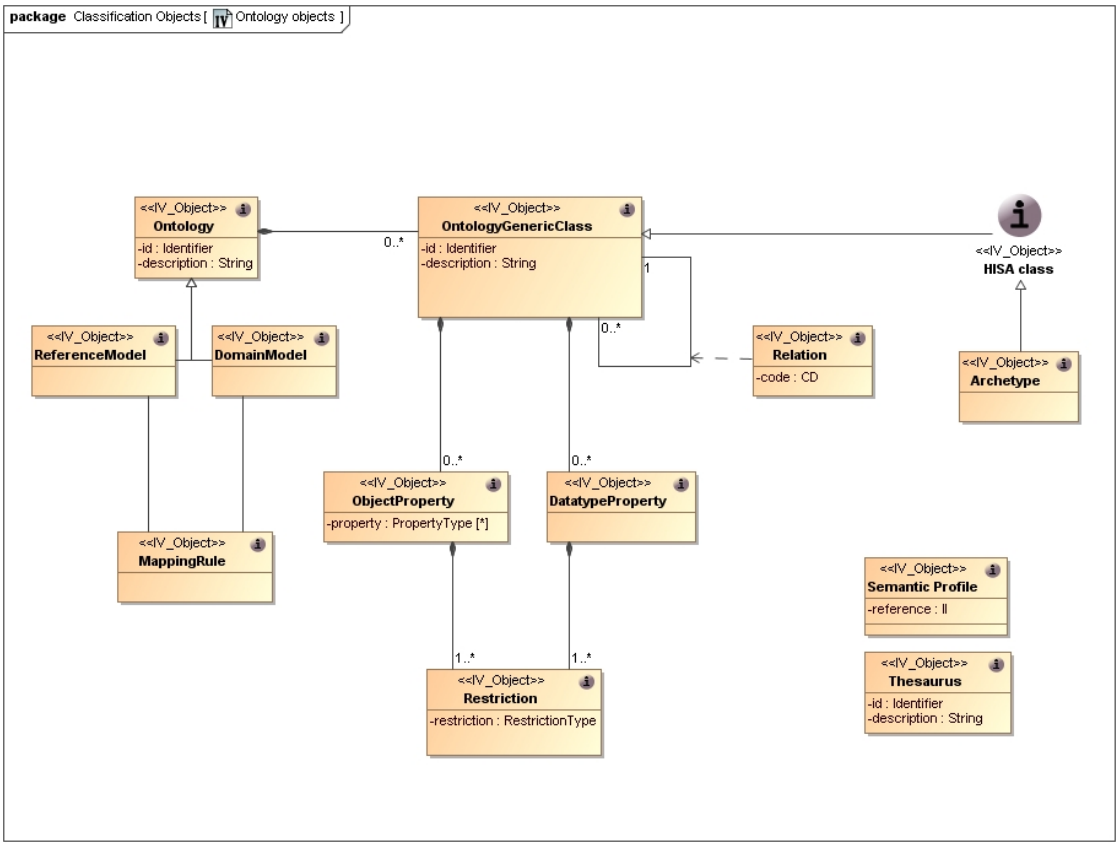

Figura 5.26. Modelo de información para los conceptos abstractos de la gestión de semántica

Las actividades de gestión de semántica permiten a los usuarios de la arquitectura introducir nuevos conceptos y relaciones, consultar las ontologías definidas, etc. Todas estas tareas son fundamentales en

un modelo federado. Si se quiere añadir un nuevo sistema a la federación necesitará establecer las relaciones de sus conceptos locales con los federados. Las técnicas de mapeo entre ontologías permiten buscar las equivalencias que existen entre conceptos de forma automática o semiautomática, y los lenguajes de ontologías expresarlas. De manera que un modelo ontológico del dominio, en lugar del tradicional modelo de información, facilita la integración de sistemas con distintos modelos.

Si se analizan los modelos de gestión semántica actualmente utilizados en las normas para el desarrollo de sistemas de gestión de historia clínica tanto en el openEHR como en la norma ISO/EN13606-2 se ha adoptado el concepto de arquetipo. Un arquetipo permite al usuario definir conceptos, en principio del dominio clínico, a partir de conceptos básicos del modelo de referencia y estableciendo restricciones sobre los mismos. Para que un componente pueda entender un arquetipo es evidente que debe conocer el modelo de referencia en el que está basado. En un modelo federado dos sistemas pueden utilizar distintos modelos de referencia de manera que es necesario relacionar los dos para encontrar las equivalencias entre ambos y que dos entidades de distintos sistemas, con distintos modelos, puedan tener una semántica común. Si los modelos de referencia y arquetipos se representan como ontologías se facilita la expresión de estas relaciones. Un arquetipo podría verse como una clase de la ontología que representa un concepto concreto modelado directamente por el usuario estableciendo restricciones sobre otras clases. De manera que en realidad la idea es muy similar salvo que mientras que los arquetipos han sido específicamente pensados para el dominio clínico, y por tanto son especialmente aptos para representar la semántica de este tipo de información, las ontologías son de propósito general. Normalizar HISA en términos de ontologías permitiría una mejor reutilización de componentes y una mayor integración con otros dominios que pueden estar implicados, hoy o en el futuro, en una organización sanitaria. Para ser totalmente compatibles con la ISO/EN13606-2, y por supuesto facilitar la incorporación a la arquitectura a todos los sistemas que utilicen la técnica de arquetipos para representar semántica, es conveniente introducir el concepto Arquetipo (*Archetype*), tal y como lo define la norma, como una clase más de la ontología (Figura 5.26).

En cuanto a la gestión de anotaciones, en este trabajo de Tesis se define una anotación (formalmente llamada *perfil semántico*) como un elemento de información ligado a algún otro elemento de información o entidad de la arquitectura y cuya finalidad es matizarlo, extenderlo o relacionarlo con otros. Hay que tener en cuenta que la anotación puede cambiar notablemente el significado de una instancia. Las anotaciones pueden ser totalmente externas al elemento de información al que se refiere o estar incluidas en él. Este segundo caso es especialmente interesante en las anotaciones sobre información en formato texto, muy habitual en la práctica diaria de la medicina. Puede resultar útil realizar las anotaciones dentro del propio texto, por ejemplo usando XML y las etiquetas definidas en algún esquema que dé significado a las mismas, aunque por motivos de seguridad podría ser necesario mantener además la versión original. En la Figura 5.26 aparece el objeto de información *Semantic profile* y como se puede ver todas las anotaciones estarán referidas a alguna instancia dentro de la arquitectura.

Otro concepto importante en la gestión de semántica es la definición de reglas las cuales, junto a la definición de la ontología y las instancias de la misma, servirán para hacer razonamientos generando nuevo conocimiento que añadir al sistema y disparar acciones de manera que pueden ser el mecanismo para el comportamiento proactivo de algunos componentes. Una regla para inferir nuevo conocimiento sigue el modelo:

*Si **Antecedente** Implica **Consecuencia***

Si en la base de conocimiento se cumplen las condiciones indicadas en el antecedente entonces también se cumplen las especificadas en la consecuencia. En el antecedente se consideran las características conocidas sobre las instancias de la base de conocimiento y en la consecuencia se indicará una nueva afirmación que se puede realizar sobre éstas. Una regla que dispare acciones sigue el modelo:

*Si **Condición** realizar **Acción***

Pero no se ejecuta en un instante predeterminado ni en un orden específico como ocurre en programación. La acción se ejecuta en el instante en el que la condición cambie de FALSA a VERDADERA. Esta condición consiste en un conjunto de patrones que se comparan con los hechos de la base de conocimiento: en este caso sería la base de conocimiento de la arquitectura sanitaria o una parte de ella. La acción es la ejecución de algo programado como por ejemplo la invocación de una operación o el envío de una señal. Como se puede apreciar un antecedente y una condición son prácticamente lo mismo. Mientras que una consecuencia implica un cambio de la base de conocimiento, una acción indica la ejecución de alguna operación que por supuesto puede introducir también algún cambio. En la Figura 5.27 se presenta un modelo simplificado de cómo se podrían expresar reglas de inferencia y de acción.

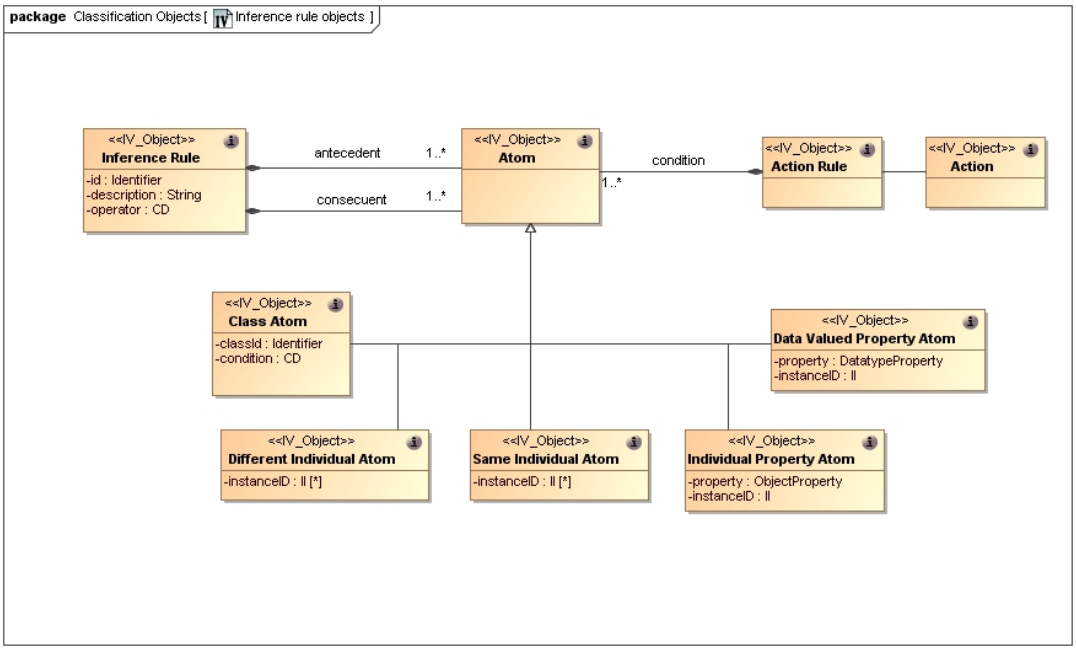

Figura 5.27. Modelo de información para las reglas de inferencia y acción

Este modelo simplificado está basado en el lenguaje SWRL [85] y en el lenguaje de expresión de reglas de Jess [83]. La conexión con el resto de las clases de la vista de la información es a través de las unidades atómicas (*atom*) donde se expresan afirmaciones sobre clases o propiedades del modelo. Las actividades de gestión de conocimiento están muy relacionadas con los razonadores que básicamente ejecutan las reglas definidas y pueden ser el soporte, por ejemplo, de aplicaciones de apoyo a la decisión clínica.

La especificación del punto de vista de la Información se completa con un esquema invariante de los objetos de información relacionados con los agentes que toman parte en los procesos de gestión de semántica. En la Figura 5.28 se muestran las relaciones entre estos objetos de información y los de la norma HISA (especificados con el círculo 'i'). Todos los objetos de información identificados en este diagrama y relacionados con componentes de gestión semántica heredan de la clase *HISA System*/*SW Component*. Se han identificado dos tipos de elementos (*Manager System* y *Provider System*) para cada elemento semántico (*Ontology*, *Semantic Profile*, *Thesaurus* e *Inference Rule*) además de un elemento encargado del razonamiento (*Reasoning System*).

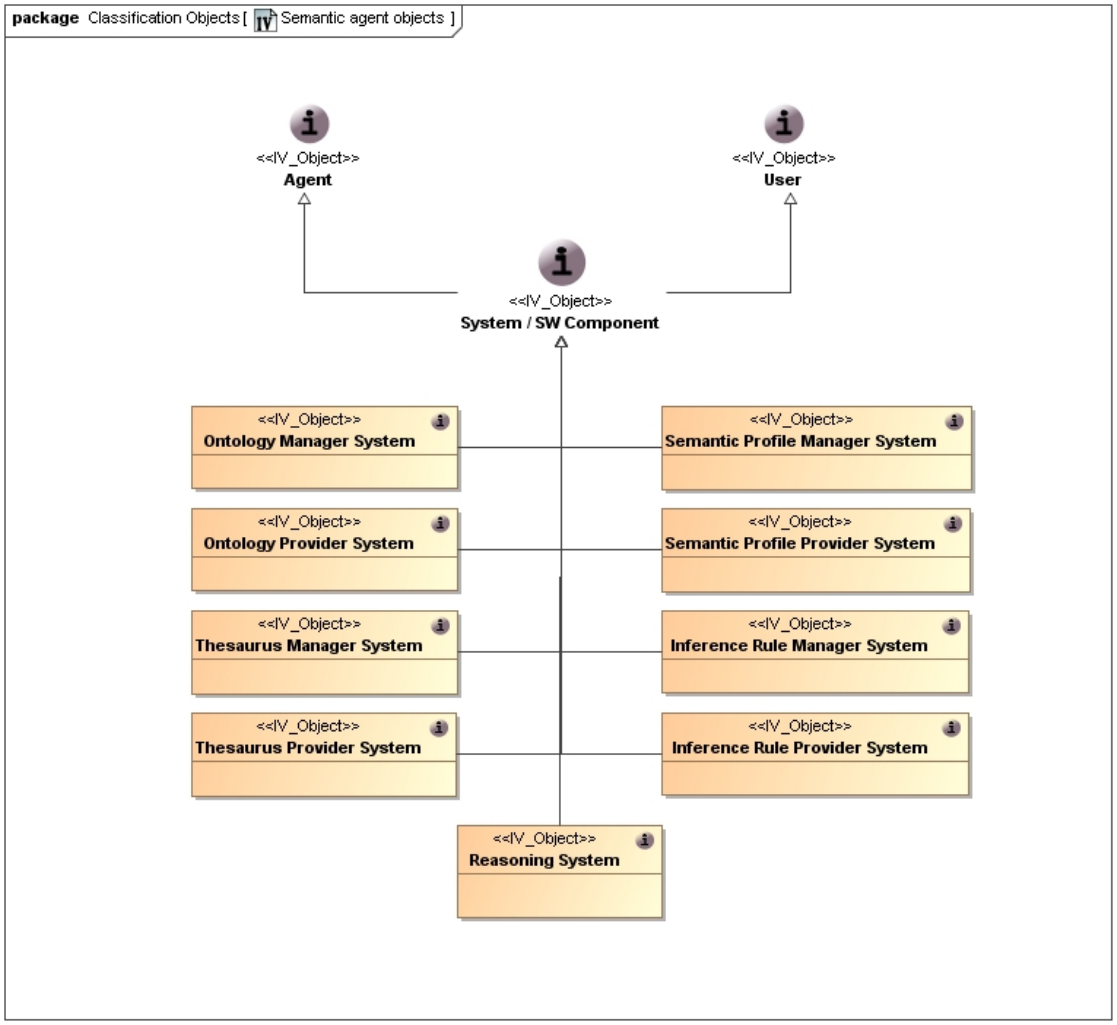

Figura 5.28. Modelo de información para los agentes soportando las capacidades semánticas

Finalmente se han identificado las acciones a partir de los procesos e interacciones definidos en el punto de vista de la Empresa. Las acciones de información se expresan en un paquete representando los tipos de acciones soportadas por los objetos de información del sistema.

5.3.3. Correspondencias entre los puntos de vista de la Empresa y la Información

Las relaciones entre los objetos de la empresa y los de la información para la gestión de semántica comprenden correspondencias entre los elementos (o artefactos) semánticos *Thesaurus*, *SemanticProfile*, *ReferenceModel*, *DomainModel*, *Archetype* y *MappingRule*; y entre los elementos agentes (el *Ontology Administration Point* con el *Ontology Manager System*, el *Ontology Information Point* con el *Ontology Provider System*, el *Thesaurus Administration Point* con el *Thesaurus Manager System*, el *Thesaurus Information Point* con el *Thesaurus Provider System*, el *Semantic Profile Administration Point* con el *Semantic Profile Manager System*, el *Semantic Profile Information Point* con el *Semantic Profile Provider System*, el *Inference Rule Administration Point* con el *Inference Rule Manager System*, el *Inference Rule Information Point* con el *Inference Rule Provider System*, y finalmente el *Reasoner Manager* con el *Reasoning System).* Paralelamente existen correspondencias entre los objetos *Ontology* e *Inference Rule* junto con las relaciones entre los procesos y acciones de información correspondientes a estos objetos.

5.4. Punto de vista Computacional

5.4.1. Formalización del punto de vista Computacional del sistema de gestión de semántica

Lo único que especifica HISA con respecto a los objetos computacionales de este grupo de actividades es que deben adherirse a la clasificación de objetos básicos o de propósito general. Además se determina que todos los objetos computacionales deben estar incluidos en un paquete separado de aquellos que pertenecerían al resto de clústeres de actividades. Al igual que se hizo con la vista de la Información, comenzamos analizando las modificaciones y extensiones propuestas en los componentes de gestión de semántica ya que son fundamentales para las tareas de integración en un modelo federado y constituyen una importante contribución del presente trabajo de Tesis Doctoral. La tarea fundamental de estos componentes es manejar las instancias de las clases relacionadas con la gestión de semántica ya presentadas en la vista de la Información y, por tanto, tratan específicamente conceptos de la ontología del dominio de la arquitectura sanitaria. En definitiva, podrían considerarse los objetos que permiten la gestión dinámica del modelo de información entre los que se podrán encontrar:

- Componentes gestores de ontologías: entre ellos estarán los que permitan modificar la ontología del dominio (esto es, el modelo de información) facilitando la introducción, borrado y modificación de clases, propiedades y restricciones. También existirán componentes que permitan consultar conceptos de las ontologías para conocer el modelo de información compartido en un determinado instante. Van a ser reutilizados por agentes especializados en el establecimiento de

las relaciones entre términos de distintas ontologías, que serán de gran apoyo a la integración. Son componentes autónomos que permiten la automatización (o semiautomatización) del establecimiento de correspondencias entre los conceptos locales a un sistema y los federados encontrando las equivalencias y diferencias entre clases y propiedades de ambas. Generalmente serán utilizados por otros componentes permitiéndoles relacionarse con terceros en términos de la ontología federada. La normalización de las interfaces de estos componentes pueden basarse en sistemas de gestión de ontologías como Jena [190] o Protégé [87] que permiten tanto la modificación como la consulta de ontologías e incluso tienen funciones de fusión ontológica. De hecho podrían reutilizarse estos componentes en la arquitectura ya que son de libre distribución.

- Componentes gestores de arquetipos: se pueden considerar un subgrupo de los anteriores, más específicos del dominio sanitario y que harán uso de los más generales. Ya que en el modelo de información se ha considerado la introducción del concepto de arquetipo tal y como se especifica en el openEHR y en la norma ISO/EN13606 es conveniente que la arquitectura cuente con agentes que los gestionen adecuadamente. Los componentes más básicos serían los que tengan interfaces para la creación, almacenamiento y/o consulta de arquetipos. Para facilitar la integración pueden existir componentes de traducción de arquetipos que permitan trasladar las restricciones impuestas en los conceptos de la ontología federada a restricciones para las ontologías locales. Asimismo pueden existir componentes que descubran el arquetipo al que se corresponde cierta instancia de información y cualquier otra gestión de arquetipos que sea necesario realizar.

- Componentes gestores de terminologías y tesauros: son los únicos considerados actualmente en HISA. La interfaz detallada puede estar basada en alguna especificación ya existente como el servicio de consulta de léxico (*Lexicom query service*, LQS) de CORBAmed. Poseerán interfaces de consulta a sistemas de términos y códigos que o bien devuelven el código correspondiente a algún concepto en determinado sistema de términos o realizan el proceso contrario. Pueden existir tareas más complejas como la traducción de un sistema a otro, la consulta de términos en un idioma específico, etc.

- Componentes de gestión de reglas y razonadores: la funcionalidad básica es soportar la creación y gestión de reglas y realizar las inferencias oportunas en base a los hechos acontecidos en el sistema. Servirán para disparar acciones en distintos componentes o para inferir nuevo conocimiento a partir del ya considerado en el sistema. En este sentido serán el soporte de componentes más complejos como los de apoyo a la decisión clínica.

- Componentes de gestión de anotaciones: las tareas de estos componentes se centran en facilitar la creación y almacenamiento de anotaciones y en manejar los vínculos con la información a la que complementan. Además se encargan de la recuperación, ayuda a la interpretación, presentación y gestión de las anotaciones. Las anotaciones matizan la semántica de la información a la que están

ligadas, la hacen más específica o incluso pueden llegar a cambiar el significado de la misma. A partir de las anotaciones se puede llegar a decidir qué componentes están más capacitados para gestionar adecuadamente la información que cualifican o cuáles pueden proporcionar un conocimiento más amplio o específico a partir de las anotaciones adjuntas. Cuando las anotaciones se refieren a observaciones clínicas se establecerán relaciones entre estos componentes y los de gestión de información clínica. En este caso las anotaciones suelen aportar conocimiento sobre la parte del cuerpo a la que afecta determinada observación, la relevancia de la misma en ciertas actividades, la actividad en la que se generó o los códigos correspondientes en distintos sistemas de clasificación.

Para facilitar el entendimiento de la especificación del punto de vista Computacional se va a exponer en primer lugar cómo se alcanza la conformidad de la misma con la norma HISA. El requisito normativo indispensable es que exista un objeto computacional por cada clase identificada en el punto de vista de la Información. Cada uno de estos objetos computacionales básicos deberá proveer con el conjunto de métodos básicos definidos en la norma. Con todo este conjunto de objetos computacionales HISA resuelve la gestión de todas las clases definidas en el modelo de información así como las propiedades e instancias de cada una de ellas.

En la Figura 5.29 se presenta el conjunto principal de objetos computacionales que dan soporte a los procesos de la Empresa vistos anteriormente. Son seis los objetos computacionales que forman el núcleo del sistema de gestión de semántica (*Ontology*, *Archetype*, *Thesaurus*, *Semantic profile, Inference* y *Inference Engine*) mientras que otros cinco dan acceso a la gestión de los diferentes elementos semánticos por parte de usuarios externos al sistema (*InterfaceToOntologyAdmin*, *InterfaceToArchetypeAdmin*, *InterfaceToThesaurusAdmin*, *InterfaceToSemanticProfileAdmin* y *InterfaceToInferenceAdmin*). Cada objeto computacional de administración está enfocado a la gestión de un tipo concreto de elemento semántico y se relaciona a través de una interfaz particular con el objeto computacional que soporta los procesos relacionados con dicho elemento. Estos objetos computacionales básicos siguen las guías especificadas por HISA en cuanto a sus métodos. Por otro lado, el sistema de inferencia posee una interfaz dedicada a la comunicación con el resto de objetos del sistema para recuperar los elementos semánticos que necesite para realizar los procesos de inferencia. Como puede verse el objeto computacional que soporta los procesos relacionados con los arquetipos está incluido dentro del objeto que gestiona las ontologías. Puesto que los arquetipos están íntimamente relacionados con las ontologías se ha optado por esta configuración de objetos computacionales. De cualquier modo las interfaces para la gestión de los arquetipos son expuestas por el sistema de ontologías pero delegadas directamente al sistema de arquetipos.

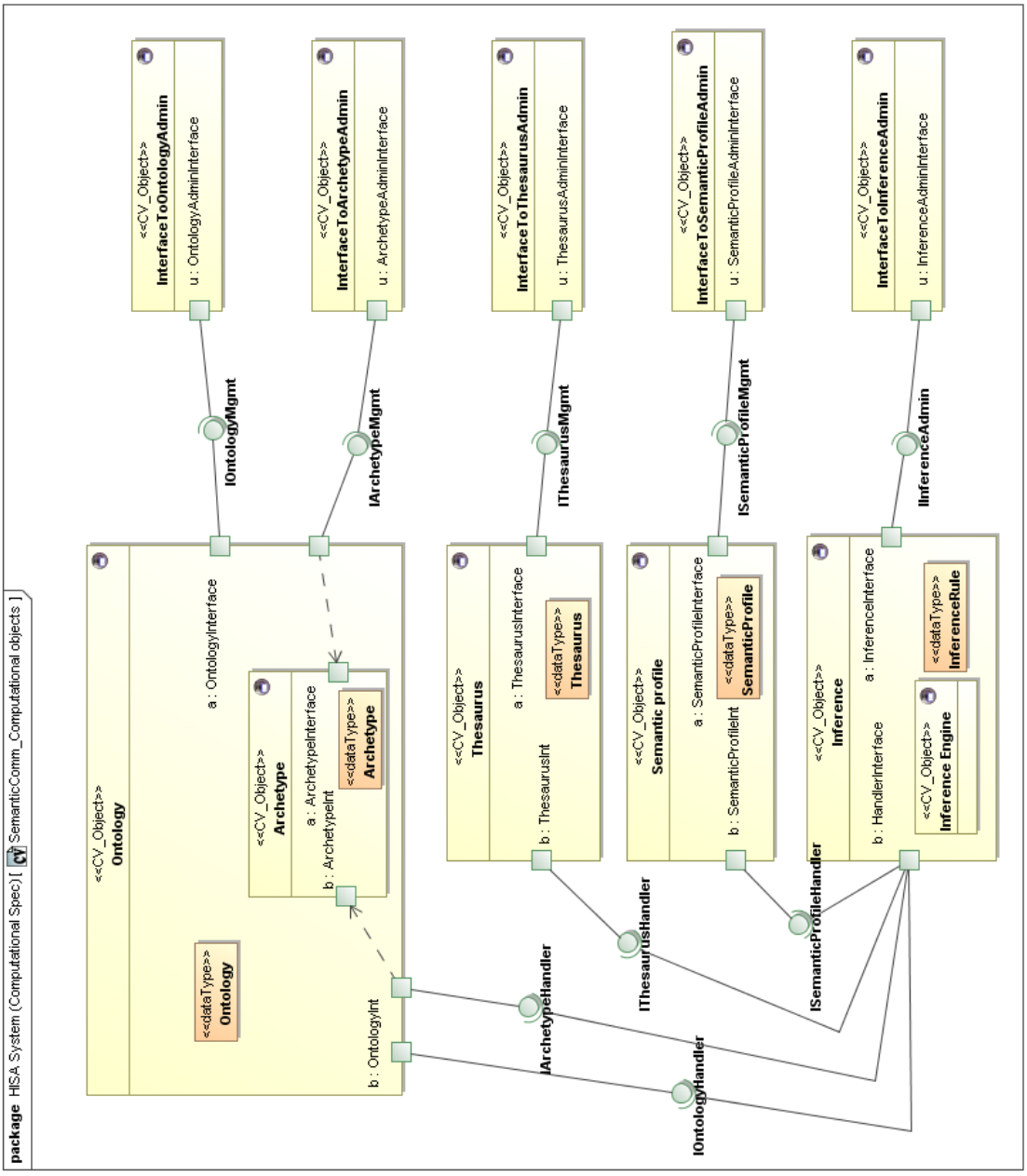

Figura 5.29. Objetos computaciones e interfaces para la gestión de semántica

Una vez descritos los objetos computacionales de este punto de vista para el sistema de gestión de semántica es necesario especificar los métodos que operan en cada una de las interfaces. Las interfaces que relacionan el sistema de inferencia con el resto de objetos computacionales (*IOntologyHandler*, *IArchetypeHandler*, *IThesaurusHandler* y *ISemanticProfileHandler*) exponen los métodos normativos de HISA para solicitar la inclusión de un elemento en el sistema (métodos de tipo "*add"*), recuperarlo (tipos "*detail"* y *"list*") o eliminarlo (métodos de tipo "*delete"* y *"pack"*). Para recuperar un elemento se envía el identificador del mismo (*OIdentifier* para ontologías, *AIdentifier* para arquetipos, *SPIdentifier* para perfiles semánticos y *TIdentifier* para tesauros). Las interfaces de gestión de elementos semánticos (es decir, *IOntologyMgmt*, *IArchetypeMgmt*, *IThesaurusMgmt* y *ISemanticProfileMgmt*) heredan de las anteriores y añaden dos tipos de métodos más: el de edición de elementos (*update*) y el de validación

(*validate*). Este último método es de tipo interrogación y el cliente lo invoca pasando el identificador del elemento a ser validado y recupera como respuesta el resultado de la validación. Además la interfaz de gestión de ontologías contiene un método (*mapping*) que permite establecer el mapeo entre dos elementos ontológicos que se le pasan como parámetros devolviendo el resultado en un tipo de dato *MappingResult*.

Finalmente, la interfaz de administración de inferencias (*IInferenceAdmin*) expone los tipos de métodos especificados por la norma HISA (*add*, *delete*, *update*, *detail* y *list*) y añade los métodos ejecutar (*execute*) y validar (*validate*) una regla de inferencia. Todos ellos siguen la misma estructura que los correspondientes al resto de elementos semánticos excepto el de ejecutar inferencia que no tiene homólogo en el resto de interfaces. El cliente envía el identificador de una regla de inferencia (*IIdentifier*) y solicita la ejecución de la misma. Recibe como resultado un tipo de dato *InferenceResult*.

Los elementos semánticos que manejan los objetos computacionales internamente (ver Figura 5.29) o en las comunicaciones entre objetos y que forman parte de los métodos se recogen como tipos de datos. Los cinco tipos de datos básicos para el sistema de gestión de semántica son: *Ontology*, *Archetype* (que hereda del anterior), *Thesaurus*, *Semantic Profile* e *Inference Rule*. Cada instancia de cada uno de estos tipos de datos tiene asignado un identificador que lo discrimina dentro del sistema. Para las respuestas de los métodos se han definido tres tipos de datos más como son:

- *Validation response:* como respuesta a todos los métodos que permiten solicitar la validación de un elemento semántico.

- *Mapping result:* es el resultado de un mapeo entre ontologías solicitado a través del método *mapping.*

- *Inference result:* contiene el resultado de un proceso de inferencia y se utiliza en el proceso *execute.*

5.4.2. Correspondencias entre los puntos de vista de la Empresa y Computacional

Los últimos puntos que quedan para completar la formalización del punto de vista Computacional del sistema de gestión de semántica son las correspondencias del mismo con los puntos de vista de la Empresa y la Información. Las correspondencias entre el punto de vista de la Empresa y el Computacional se concretan en base a dos tipos de elementos: los procesos y los objetos. Las correspondencias más relevantes son:

- entre artefactos de la empresa y tipos de datos computacionales: aquí encontramos los objetos *Ontology, Thesaurus, Inference Rule, Archetype* y *Semantic Profile;*

- entre objetos de la empresa y objetos computacionales: donde se encuentran los pares *Ontology Information Point/Ontology System, Thesaurus Information Point/Thesaurus System, Ontology*

Administration Point/InterfaceToOntologyAdmin, Thesaurus Administration Point/InterfaceToThesaurusAdmin, Semantic Profile Administration Point/InterfaceToSemanticProfileAdmin y *Semantic Profile Information Point/Semantic Profile System.*

En el último grupo faltan dos tipos de objetos de empresa y computacionales, en concreto, los relacionados con los arquetipos y con las inferencias. La explicación es diferente en cada caso. Por un lado, el objeto de empresa *Archetype* fue tenido en cuenta en la formalización del punto de vista de la Empresa aunque sólo a título informativo y no se especificaron los procesos asociados a ese objeto como correspondería de acuerdo con el tratamiento que se le ha dado al resto de elementos semánticos. De igual forma ocurrió en la formalización del punto de vista de la Información donde el objeto de información *Archetype* heredaba de *HISAClass*. Es en el punto de vista Computacional donde se concreta el sistema para la gestión de arquetipos dentro del sistema de ontologías. Debido a esto los objetos computacionales que operan sobre arquetipos no tienen correspondencia con objetos de empresa o de información.

Por otro lado, existen tres objetos de empresa (*Inference Rule Administration Point*, *Inference Rule Information Point* y *Reasoner Manager*) y tres objetos computacionales (*Inference System*, *InterfaceToInferenceAdmin* y *Inference Engine*) relacionados con la gestión de inferencias. La correlación no puede darse uno a uno puesto que los objetos de la empresa están separados mientras que el objeto computacional *Inference Engine* está incluido en el objeto *Inference System*. Aunque las funcionalidades de ambos grupos de objetos se corresponden no sería adecuado obligar la relación de uno a uno así que se opta por dejar el motor de inferencia como parte del sistema de inferencia pero sin concretar las interfaces que expone. Será el sistema de inferencia el que soporte la ejecución de inferencias ya sea delegando en el motor o utilizándolo como un componente que forma parte de él mismo.

5.4.3. Correspondencias entre los puntos de vista de la Información y Computacional

Las correspondencias entre el punto de vista Computacional y el de la información se pueden agrupar en dos grupos:

- Relaciones de objetos de información con los tipos de datos computacionales: aquí encontramos los objetos *Ontology, Thesaurus, Semantic Profile, Inference Rule* y *Archetype*, homónimos en ambos puntos de vista.

- Relaciones de objetos de información con objetos computacionales: destacando *Thesaurus Manager System* con *InterfaceToThesaurusAdmin, Ontology Provider System* con *Ontology System, Thesaurus Provider System* con *Thesaurus System, Ontology Manager System* con *InterfaceToOntologyAdmin, Semantic Profile Manager System* con

InterfaceToSemanticProfileAdmin, Semantic Profile Provider System con *Semantic Profile System* y *Reasoning System* con *Inference Engine.*

En la especificación de correspondencias entre estos puntos de vista encontramos los mismos aspectos a destacar que en el caso de las correspondencias entre el punto de vista de la Empresa y el Computacional, es decir, la falta de los elementos relacionados con la gestión de arquetipos y con la gestión de inferencias. El razonamiento de esta ausencia de correspondencias es el mismo que el mostrado en el apartado anterior.

Aparte de estos grupos de correspondencias deben especificarse las relaciones que estipula la norma HISA entre los objetos de información y aquellos objetos computacionales básicos que agrupan los métodos de gestión de los primeros pero para todos y cada uno de los objetos de información identificados. Estas relaciones no se muestran aquí puesto que esa correspondencia entre objetos, aunque existe, no sigue el propósito de los *CorrespondenceLinks* cuyo objetivo es identificar el mismo componente del sistema en dos puntos de vista separados. Por ello se entiende que las relaciones entre las clases de información y los objetos computacionales gestores de instancias de esas clases no son objeto de este tipo de correspondencias.

6. Resultado 5. *Person-Oriented Virtual Organization* (POVO)

6.1. Introducción

Con la evolución de las TICs y su aplicación en sanidad cada vez se dispone de mayor información y recursos dedicados a cada Sujeto de Asistencia (SdA). Surgen entonces propuestas que orientan los recursos y procesos asistenciales hacia escenarios que sustituyen la organización sanitaria por el SdA como elemento central. Este tipo de escenarios potencian la personalización de tratamientos y procesos a las necesidades concretas de cada SdA e incrementan su implicación en su propia asistencia sanitaria (por ejemplo expresando intereses y preferencias, participando en la toma de decisiones médicas [191], animando al aprendizaje y la autogestión a lo largo de su vida, etc.). Asimismo se podría entender la asistencia sanitaria más allá del tratamiento puntual de una dolencia incluyendo la prevención de enfermedades, el seguimiento y la mejora de la calidad de vida. Dicho modelo sanitario presenta también una enorme complejidad debido, por ejemplo, a las necesidades de coherencia en la integración de toda la información y conocimiento sobre un SdA en cualquier momento y lugar [192]. En este apartado se presenta la formalización de una arquitectura abierta de soporte a escenarios centrados en el SdA. Dicha arquitectura hereda las contribuciones de los apartados anteriores con respecto al uso de HISA como estándar para la formalización de arquitecturas sanitarias junto con las extensiones realizadas tanto en control de acceso como en gestión de semántica. Esta arquitectura va a soportar el escenario de organización virtual centrada en la persona (*Person Oriented Virtual Organization, POVO*) que hace hincapié en la definición de una organización virtual cuyo objetivo es el mantenimiento de la salud de un SdA.

En el ámbito de los sistemas sanitarios centrados en el SdA (y para nuestro escenario POVO) un punto de discusión es la propiedad de la información de los sujetos y quién tiene autoridad para decidir las políticas de acceso a ella. Varios países han promulgado leyes [193][194] (y el conjunto de Europa, recomendaciones [195-197]) apoyando la idea de que cada individuo debe ser capaz de controlar la información y los recursos relacionados con su salud evitando accesos no autorizados. La tendencia es involucrar al SdA no sólo en el mantenimiento de su salud (a través de la toma de conciencia de toda su información y recursos) sino también en la gestión y control de acceso a ellos mediante el establecimiento de criterios que él considere adecuados. Este modelo de gestión de recursos sanitarios donde el SdA es el administrador de las políticas de control de acceso a sus recursos y los sistemas deben ajustarse a sus preferencias no es fácilmente alcanzado con los sistemas desplegados en la actualidad. Si se consideran aspectos de integración y distribución lograr este objetivo es incluso más difícil. Actualmente se pueden encontrar iniciativas que intentan acercar la gestión de los recursos sanitarios al individuo al cual se refieren como el Registro Personal de Salud (*Personal Health Record, PHR*) [198], el Registro de Salud Controlado por la Persona (*Person Controlled Health Record, PCHR*) [199], los escenarios con tarjetas inteligentes (*smart-cards*) [200] u otros [201][202]. Aunque estos ejemplos apuestan por modelos sanitarios centrados en el SdA, la mayoría proponen escenarios centralizados donde los recursos pertenecen a un único dominio administrativo. El potencial de reutilización de recursos desplegados en otros dominios es ignorado o subestimado principalmente a causa de la complejidad que la distribución de elementos añade. Los escenarios sanitarios con recursos distribuidos no son una propuesta de futuro ya que actualmente cualquier SdA tiene recursos (información, dispositivos dedicados, etc.) relacionados con él en diferentes organizaciones sanitarias a través de regiones y países diferentes. Una propuesta realmente centrada en el SdA debería ser transparente a las localizaciones geográficas y administrativas de los recursos y operar en cualquier dominio que mantenga recursos relacionados con el SdA.

Intentando cumplir con estos requisitos de distribución de componentes y control de acceso el escenario POVO presenta una enorme complejidad ya que aúna recursos y sistemas sanitarios dentro de una organización virtual que tiene un carácter dinámico y puede evolucionar a lo largo del tiempo. Para la formalización de la arquitectura de soporte de la POVO se adopta la propuesta del RM-ODP:

- Puntos de vista independientes de tecnología (Empresa, Información y Computacional).

  La formalización de las tres vistas independientes de tecnología descrita en apartados anteriores (esto es, la especificación de la norma HISA junto a sus extensiones de seguridad y gestión de semántica) se toma como base y se amplía para acomodar los requisitos específicos del escenario POVO. De esta forma la arquitectura sanitaria de referencia será adaptada a las necesidades de un caso particular que en este trabajo de Tesis Doctoral es el paradigma POVO. Además se combinarán las extensiones de seguridad y semántica para mejorar el sistema de control de acceso descrito beneficiándose de las capacidades de gestión semántica.

- Punto de vista de Ingeniería.

  La especificación de ingeniería define la infraestructura necesaria para soportar la distribución de un sistema ODP identificando las funciones requeridas para gestionar la distribución física, comunicación, procesamiento y almacenamiento. En lugar de desarrollar una infraestructura de distribución propia se utilizará el estilo arquitectural SOA para definir el modelo de despliegue de elementos y funciones. Se añadirán asimismo algunas características de estilo arquitectural de la computación en Grid que comparte con SOA la misma base teórica.

- Punto de vista de Tecnología.

  Finalmente, el punto de vista de Tecnología se centra en la elección de la tecnología concreta para implementar el sistema ODP. En este punto de vista se concretarán los objetos de los distintos puntos de vista en elementos tecnológicos específicos ya sean heredados de otras iniciativas o desarrollados *ad hoc* para el escenario POVO.

En este trabajo se ahonda en un aspecto clave del escenario POVO como son los mecanismos de control de acceso basados en semántica y centrados en el SdA. Se ha hecho especial hincapié en las técnicas de especificación de políticas de control de acceso permitiendo al SdA gestionar el acceso a sus recursos sanitarios de una manera amigable y con gran flexibilidad. El sistema de control de acceso que se presenta se diferencia de otras contribuciones en dos puntos principales. Por un lado, está completamente orientado a ser parte de una arquitectura sanitaria de referencia por lo que tiene en cuenta la armonización de estándares e iniciativas en este campo además de la formalización del sistema en puntos de vista separados. Por otro, se ha potenciado la usabilidad de las soluciones para facilitar las tareas de administración al SdA. Un ejemplo de esto es el desarrollo de la aplicación M3A (*Me-As-An-Admin*) que guía al SdA a través de la especificación de políticas de control de acceso a sus recursos distribuidos mediante tecnologías semánticas.

6.2. El paradigma POVO: vistas independientes de tecnología

6.2.1. Introducción

El modelo centrado en el SdA debe considerar un conjunto de recursos (humanos, administrativos, computacionales y de información) pertenecientes a diferentes dominios administrativos y tecnológicos, a menudo geográficamente separados. Además los recursos tienden a estar sujetos a reglas de compartición definidas por un administrador que en este caso es el propio SdA al que los recursos se refieren. Esta definición comparte algunos aspectos con el concepto de organización virtual (*Virtual Organization, VO*) [69] desarrollado originariamente para los procesos de negocio y luego adoptado por las arquitecturas de sistemas distribuidos [203][204]. El modelo de asistencia sanitaria centrado en el SdA puede enlazarse con el concepto de VO determinando que el objetivo común de los recursos que

motiva la colaboración es la asistencia sanitaria, social y del bienestar de un SdA particular (que es el administrador de la VO). Para remarcar esta orientación al SdA se propone un nuevo paradigma específico para este dominio: la organización virtual orientada a la persona (o por su acrónimo en inglés, POVO). Tanto la VO como la POVO integran sistemas heterogéneos distribuidos a través de fronteras administrativas y los diferentes dominios deben definir enlaces de cooperación entre ellos. Una importante diferencia entre ambas propuestas es que la seguridad en una VO es compartida entre los administradores de los dominios implicados mientras que en una POVO el ciudadano correspondiente es el administrador exclusivo de los recursos y puede decidir sobre el acceso a los mismos. Además, mientras una VO es dinámicamente creada para llevar a cabo un proceso de negocio, una POVO está fuertemente unida al proceso de asistencia sanitaria y social realizado durante toda la vida de una persona particular. De esta forma, cada individuo tiene una única POVO dedicada que evoluciona para ajustarse a sus deseos y es disuelta cuando el sujeto fallece (Figura 5.30).

A pesar de las aparentes similitudes entre VO y POVO, las soluciones tecnológicas que permiten desplegar una organización virtual [203-205] presentan mecanismos de seguridad inadecuados para el dominio sanitario debido a que la información que se maneja en éste es considerablemente más sensible. La protección e integridad de los datos deben ser aseguradas de forma rigurosa por tanto el concepto VO no es suficientemente completo para cubrir todos los requisitos e implicaciones demandadas por la gestión de recursos en escenarios sanitarios. Esta es otra de las razonas por las que se propone el nuevo concepto POVO. El propósito de utilizar el término *persona* (en lugar de *paciente*) es remarcar que el escenario POVO se centra en SdAs los cuales no son sólo pacientes en tratamiento o monitorización sino también cualquier individuo sano implicado en tareas de prevención siguiendo el modelo de continuidad de la asistencia. Así, ciudadanos sanos pueden, por ejemplo, gestionar recursos que le ayuden a mantener hábitos de vida saludables (ingestión de comidas, ejecución de ejercicios, etc.), mantener su bienestar, prevenir enfermedades a las que están predispuestos genéticamente, etc.

El complejo modelo propuesto tras el concepto POVO conlleva nuevos requisitos y escenarios a estudiar. Entre el conjunto de requisitos generales están aquellos inherentes a la distribución de los componentes como la identificación unívoca de los elementos, el descubrimiento de los mismos, comunicaciones seguras, autenticación a través de diferentes dominios administrativos, etc. Debido a que el escenario POVO es soportado por la arquitectura abierta especificada anteriormente, se consideran resueltas las transparencias de distribución por la infraestructura de soporte y la tecnología que se utilizará (ver especificación de los puntos de vista de Ingeniería y Tecnología). El mecanismo de control de acceso sí va a ser desarrollado para el escenario POVO ya que se considera uno de los bloques constructivos principales del escenario. A continuación se presenta el conjunto de requisitos específicos de autorización que deben ser cubiertos de forma satisfactoria por las soluciones desarrolladas.

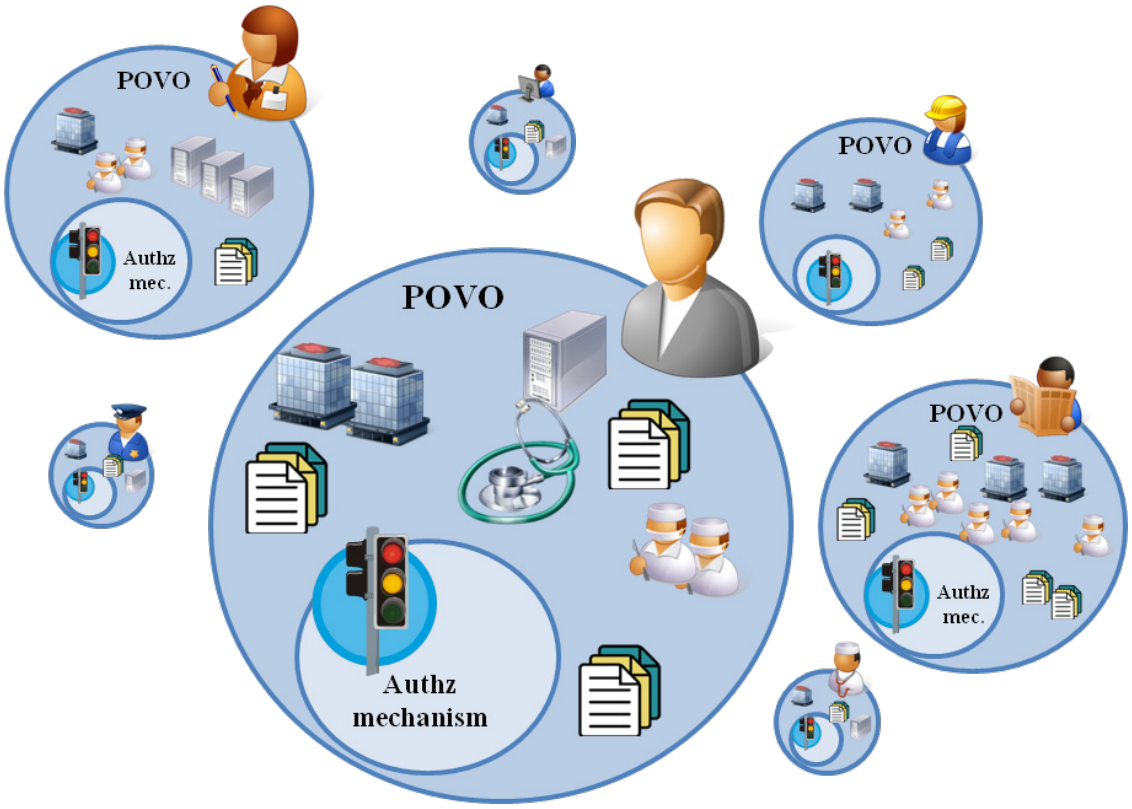

Figura 5.30. Cada persona disfrutará de una POVO agrupando los recursos relacionados con ella

1. El paradigma POVO está centrado en el SdA por lo tanto habrá un único ejemplar POVO para cada persona. Esta POVO cubrirá todos los recursos relacionados con el SdA a lo largo de toda su vida, evolucionando en cualquier momento para ajustarse a los deseos del ciudadano.

2. El SdA es el propietario de toda la información relacionada con su salud y debe tener autoridad sobre el acceso a esa información dentro de su POVO. Al margen de esto, el SdA sólo debe ser capaz de controlar el acceso a los recursos de su POVO cuando éstos refieran información sobre él.

3. Las políticas de control de acceso deben ser tan flexibles como sea posible y permitir diferentes niveles de granularidad cubriendo el rango más amplio posible de deseos y preferencias del SdA.

4. El SdA debe ser capaz de crear, modificar y eliminar sus políticas de control de acceso.

5. Tanto la interfaz de gestión como las terminologías utilizadas deben facilitar la interacción del usuario final (el SdA) y promover la usabilidad. La complejidad de los sistemas subyacentes a las tareas de administración debe ser tan transparente como sea posible para así permitir que el SdA pueda explotar al máximo las capacidades de gestión de su POVO sin necesidad de conocimiento avanzado o experto.

6. Deben existir mecanismos de delegación de privilegios de gestión para cubrir escenarios donde el SdA no desee gestionar sus recursos. Además, dichos mecanismos deben ser suficientemente

flexibles para soportar la administración por parte de los padres o tutores legales en caso de menores o discapacitados.

7. La legislación regional así como directivas internacionales [193-197] establecen escenarios donde la autoridad del SdA sobre sus recursos puede ser temporalmente invalidada. Por ello acceder a un recurso sin el consentimiento del SdA y violar las políticas existentes debe ser posible si, por ejemplo, existe un riesgo para la salud pública o una emergencia que pueda resultar en daños irreversibles o la muerte del SdA u otros si no se actúa de forma inmediata.

8. Como los recursos serán desplegados y mantenidos por organizaciones sanitarias o terceros, estarán sujetos a las políticas de acceso (y uso) definidas por sus gestores. Si esos mismos recursos refieren información relacionada con el SdA (por ejemplo, un modelo computacional de una compañía de seguros que recibe como entrada datos sanitarios de un individuo) estas políticas no pueden interferir con las especificadas por el SdA para controlar el acceso a los recursos dentro de su POVO (es decir, implicando información relacionada con su salud).

9. Deben existir mecanismos de auditoría apoyados en el registro de actividades por los usuarios en un orden cronológico. Debe registrarse toda la información sobre intentos de acceso (exitosos o no), uso o modificación de los recursos.

Estos requisitos cubren los aspectos de control de acceso requeridos en los escenarios normales de la POVO. Existen además complejas cuestiones que deben ser analizadas y resueltas al menos de forma teórica. En lo que resta se presentan estas cuestiones junto con el análisis de cómo la tecnología podría soportarlas. La solución tecnológica que se presenta a continuación es independiente de las decisiones tomadas en estos casos excepcionales y podría soportar otro tipo de políticas. Algunos casos excepcionales como los presentados han sido considerados y resueltos como prueba de concepto pero cualquier caso (discutido aquí o no) deberá obedecer la legislación vigente en el momento de su uso. El primer escenario que se debe analizar es aquel en el que el SdA es menor de edad. En el escenario general los mecanismos de delegación de derechos de administración permiten, si el SdA así lo desea o está legalmente incapacitado, otorgar los privilegios de la gestión de su POVO a otra persona o entidad administrativa. Desde ese momento el delegado obtiene el papel de administrador de la POVO y es capaz de acceder a la información y los recursos así como crear, modificar y borrar políticas como si él fuera el propio SdA. Pese a ello en ningún momento el SdA pierde sus derechos y puede continuar ejerciéndolos. El caso de menores de edad posee complicaciones adicionales porque no es una situación permanente como la anterior. Ahora el escenario sufre una evolución ya que, en primer lugar, la gestión de la POVO es realizada por los padres o tutores legales y el menor no puede acceder a su información y recursos. Cuando el menor alcanza la edad legal se hace responsable de la administración de su POVO y los padres y tutores pierden sus privilegios. Incluso este proceso puede suceder de forma gradual (conforme a la legislación vigente) y a medida que el menor crece más privilegios recibe para gestionar

aspectos de su información sanitaria, normalmente comenzando por los elementos menos sensibles. La especificación de políticas propuesta en este trabajo permite establecer límites en el acceso a la información para cada individuo (el SdA y su tutor) durante este proceso. Cuando el menor alcanza la mayoría de edad se deben eliminar todas las políticas que otorgaban al tutor acceso a los recursos del SdA.

El segundo escenario que merece ser considerado es aquel en el que varias POVOs coexisten y confluyen. El SdA es el propietario de toda la información relacionada con su salud y dentro de su POVO él es la máxima autoridad que controla el acceso a los recursos que contienen o referencian esa información. Así el acceso a la información de cualquier naturaleza puede ser restringido a los individuos que el SdA desee. El conflicto surge cuando el acceso a cierta información es denegada a un individuo que está implicado en ella. Por ejemplo, un padre con una enfermedad hereditaria establece políticas para prohibir el acceso a esa información a sus hijos. En este caso dos POVOs (y los derechos asociados) entran en conflicto porque el padre tiene derecho a ocultar su información sanitaria a quien desee (incluido su hijo) mientras que el hijo tiene derechos legales a conocer y gestionar toda la información sobre su salud (siempre dentro de un escenario de mayoría de edad), incluyendo aquella de naturaleza hereditaria que pertenece a su padre. Cualquier solución invade los derechos de uno de estos sujetos. En el mecanismo de control de acceso propuesto en este trabajo se ha dado un marco totalmente flexible que puede acomodar las decisiones tanto a favor de uno de los sujetos como del otro. Como ejemplo ilustrativo en la prueba de concepto hemos resuelto que el derecho de cualquier SdA de conocer toda la información relacionada con su salud tiene mayor prioridad que el derecho de mostrar u ocultar esa información a otros. La implementación de esta restricción se lleva a cabo a través de la política legislativa siguiente (ver Apartado 6.4.2 para la descripción de las políticas de control de acceso):

$$\text{who:Person}(?per) \wedge \text{what:Clinical\_Information}(?inf) \wedge \text{attr:Identifiable\_Subject}(?id) \wedge \text{isRelatedTo}(?inf, ?id) \wedge \text{isSubject}(?per, ?id) \rightarrow \textbf{actionPermitted}(?per, ?inf)$$

El tercer escenario se centra en la autoría de la información sobre un SdA. La información es obtenida de diferentes fuentes como dispositivos de monitorización, resultados de pruebas de organizaciones sanitarias, información genómica de laboratorios, diagnósticos y tratamientos prescritos por profesionales sanitarios, datos demográficos, hábitos alimenticios y rutinas de ejercicio introducidos por el propio SdA u otros usuarios. Toda esta información está relacionada con el SdA y por ello tiene autoridad para restringir el acceso a ella. La cuestión es si el autor de un elemento de información (por ejemplo un médico que ha introducido un diagnóstico y observaciones personales sobre el SdA) debe ser capaz de acceder a esa información en el futuro o, por el contrario, si el SdA puede obviar la autoría de la información y denegar el acceso al propio autor. En este trabajo se resuelve esta cuestión permitiendo siempre a un individuo acceder a la información que ha creado aunque el SdA decida restringir el acceso. Se ha considerado que un profesional sanitario debería ser capaz de revisar sus diagnósticos, observaciones, prescripciones, etc., como un aspecto clave de la continuidad de la asistencia y seguimiento de los SdAs.

6.2.2. Descripción del mecanismo semántico de control de acceso de la POVO

El mecanismo de control de acceso que se propone para el paradigma POVO está basado en dos pilares fundamentales: la distribución y la gestión de semántica. Todos los componentes del mecanismo de control de acceso se constituyen como elementos autónomos que publican sus capacidades y pueden ser invocados dentro de un escenario distribuido entre dominios geográficos y administrativos. Esta capacidad, unida a la adopción de estándares de comunicación, permite desplegar un paradigma de gran flexibilidad y dinamismo donde múltiples elementos independientes (redundantes o no) pueden integrarse o eliminarse del sistema sin menoscabar las capacidades de control de acceso del mismo. Por otro lado, la inclusión de semántica en los procesos de autorización permite el manejo de una ontología de conceptos controlada (con mapeo a modelos locales según el modelo federado) que facilita la interoperatividad e integración de elementos en el sistema y además permite la automatización de los procesos de toma de decisiones de control de acceso e inferencia de información sobre recursos, usuarios y entorno. En la Figura 5.47 se muestra el esquema de control de acceso particularizado para las tecnologías escogidas.

La infraestructura de seguridad para las POVO está basada en los estándares y recomendaciones de seguridad revisados anteriormente y armonizados y formalizados en la arquitectura distribuida abierta. Hasta este punto lo especificado sobre el mecanismo de control de acceso tiene en cuenta solamente aspectos independientes de tecnología y la resolución tecnológica del mecanismo está basada en dos puntos principales:

1. La herencia y reutilización de soluciones existentes de código abierto para cada uno de los componentes que forman parte del mecanismo de control de acceso. Obviamente estas herramientas deben ser alineadas con la filosofía de apertura e interoperatividad de la arquitectura, por tanto será necesario adaptarlas para su conformidad con los estándares de comunicaciones y del dominio sanitario utilizados.

2. Debido a las ventajas en interoperatividad, apertura e integración que aportan las tecnologías semánticas, se ha considerado que la resolución tecnológica del mecanismo se apoye en capacidades de gestión de semántica.

El esquema de alto nivel del mecanismo de control de acceso armonizado reproducido en la Figura 5.11 se presenta en la Figura 5.31 particularizado con elementos de gestión semántica. En concreto se presentan tres contribuciones principales:

- Los proveedores de información (de usuarios, recursos, entorno, etc.) ahora hacen uso de bases de conocimiento semánticas. Esto se traduce en que manejan ontologías con clases e instancias y pueden ejercer procesos de inferencia sobre las mismas para extraer conocimiento implícito. En el escenario POVO el número de proveedores de información es indeterminado y cada base de conocimiento podrá almacenar ontologías locales y/o la ontología federada. El mapeo entre

ontologías podrá ser realizado por los proveedores, por el gestor de contexto o el agente de decisión. El mapeo de ontologías y los procesos de inferencia generales (fuera de los realizados para la toma de decisiones de control de acceso) pueden ser delegados a la propia arquitectura y sus mecanismos de gestión de semántica que se formalizaron en los puntos de vista independientes de tecnología.

- Las políticas de control de acceso se basarán en los conceptos de las ontologías y, por tanto, se utilizarán reglas semánticas que puedan ser aplicadas por un razonador sobre una ontología de clases y sus instancias.

- El agente de decisión contará con un razonador encargado de realizar procesos de inferencia para soportar la toma de decisión de acceso.

En esta propuesta se utiliza una ontología básica como ontología federada que cubre los principales aspectos de la infraestructura de la POVO y el mecanismo de control de acceso. Aunque dicha ontología se describirá más adelante en detalle, cabe destacar aquí que los atributos potenciales que se pueden atribuir a los usuarios están contenidos en la misma y permiten determinar su relación con el SdA, el cual además es el administrador de su POVO. Este aspecto resuelve el problema de implementar modelos basados en roles (RBAC) en escenarios multiorganizacionales con fronteras administrativas separadas donde los roles están relacionados con la jerarquía dentro de las diferentes organizaciones. En tales escenarios el mismo rol en dos organizaciones podría no corresponderse con el mismo conjunto de atributos o dos roles diferentes en dos organizaciones referirse al mismo conjunto de privilegios. En este trabajo se utiliza la relación de los usuarios con el SdA porque este es un concepto que permite tener roles independientes de jerarquías administrativas. Al margen de esto, y como se explica en el apartado correspondiente, se ha introducido en la ontología una muestra del esquema de roles estructurales y funcionales especificados por la norma ISO 21298 y adoptados por la ISO 22600. De esta forma puede siempre recurrirse a este esfuerzo normalizado para caracterizar a los usuarios en función de los roles que desempeñan en las organizaciones sanitarias.

Tres principales ventajas pueden obtenerse de utilizar ontologías para describir recursos y motores de inferencia para el razonamiento. Primero, se potencian las características que vamos buscando de apertura e interoperatividad de sistemas distribuidos ya que el entendimiento entre las diferentes partes y componentes implicados es más sencillo utilizando vocabularios controlados y el mapeo entre ontologías locales y federadas. Las ontologías serán el mecanismo de definición formal de descriptores de recursos y el administrador utilizará las clases de las ontologías para etiquetar los recursos de su POVO. La segunda ventaja es que al pasar una ontología de conceptos (y sus correspondientes instancias) a través de un razonador puede inferirse nuevo conocimiento sobre los recursos y añadirse como relaciones y elementos explícitos. Finalmente, al introducir inferencia semántica en los mecanismos de control de acceso, el desarrollo de los elementos que toman decisiones se simplifica ya que las políticas de control de acceso serían expresadas conforme a las ontologías (esto es, recursos,

atributos de usuarios, aspectos de entorno, etc.) y utilizando lenguajes de reglas. Así los elementos con la lógica de decisión podrían ser reducidos a motores de inferencia y los resultados de sus razonamientos serían la decisión de permiso o prohibición de acceso.

Las políticas de control de acceso tomarán la forma de reglas procesables por los sistemas de forma automática y con base en las ontologías desarrolladas. Pero un requisito esencial para hacer viable la gestión por parte de cualquier SdA de su POVO es potenciar la usabilidad y la transparencia de los detalles de bajo nivel. Entre las tareas de administración el SdA debe ser capaz de definir políticas de control de acceso en diferentes niveles de concreción. Un ejemplo de política generalista es "permitir el acceso a los recursos de mi POVO a cualquier profesional sanitario" mientras que una política específica podría ser "permitir a mi pareja ver toda mi información relacionada con enfermedades de transmisión sexual desde el año 2000 y en la que no aparezcan terceras personas". En la prueba de concepto realizada sobre mecanismos de autorización en el paradigma POVO se ha desarrollado una aplicación de usuario final para traducir las preferencias del SdA a reglas semánticas procesables. Esta aplicación guiará a la persona a través de la especificación de las políticas de control de acceso y ocultará todos los detalles de bajo nivel.

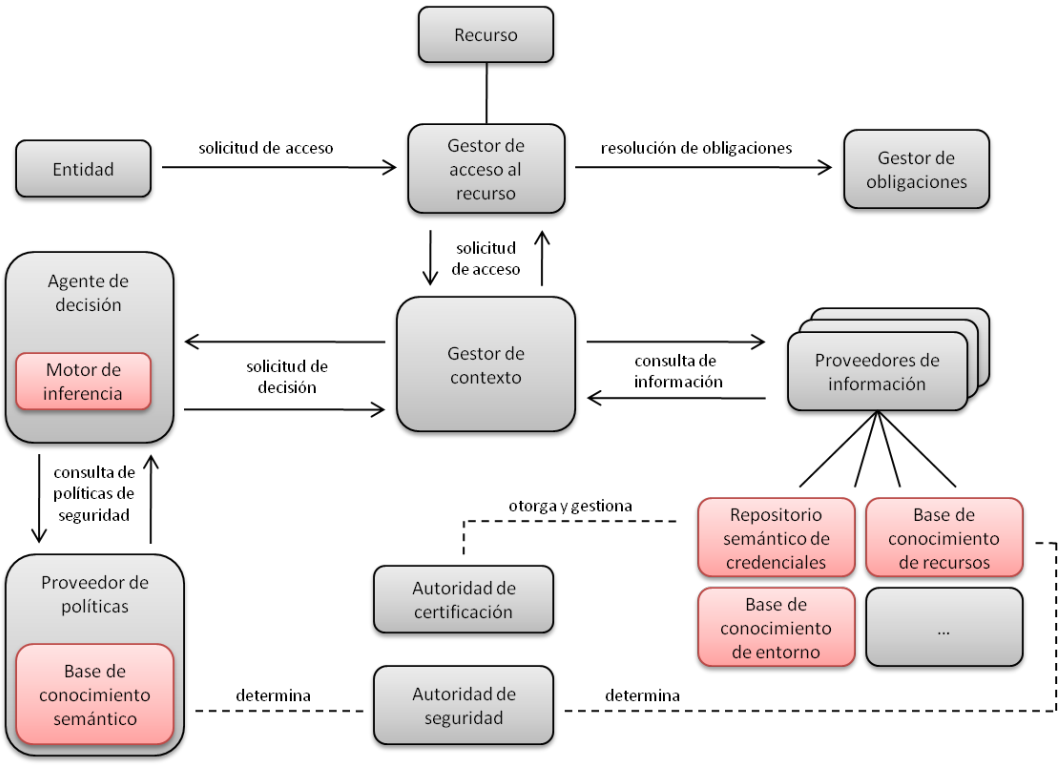

Figura 5.31. Modificaciones semánticas al mecanismo de control de acceso de POVOs

6.2.3. Extensiones de la arquitectura sanitaria para acoger el paradigma POVO

La arquitectura de servicios sanitarios formalizada hasta el momento tiene tres componentes principales de interés en el presente trabajo: la comunidad *HIS* (que aúna los aspectos normalizados del estándar

ISO12967), la sub-comunidad *Security Infrastructure* (recogiendo los aspectos de seguridad en general y de la disciplina de autorización en particular) y la sub-comunidad *Semantic Management* (para todos los elementos relacionados con la gestión de semántica). Tanto la comunidad HIS como la sub-comunidad de gestión de semántica han sido definidas de forma completamente genérica y común a todos los sistemas que puedan pertenecer a la arquitectura por ello la formalización de los puntos de vista para ambas especificaciones permanece inalterada pues encaja de manera flexible con el escenario POVO. El único cambio susceptible podría ser la modificación del objetivo de la comunidad *HIS.* Actualmente el objetivo es "*to support health care, organizational and managerial activities within a health care organization*" el cual puede refinarse para incluir el paradigma POVO como "*to support health care, organizational and managerial activities focused on a Subject of Care across administrative and organizational boundaries*".

Al margen de este punto es la sub-comunidad de seguridad la que debe sufrir ligeras modificaciones para encajar con el escenario POVO. Por un lado se considera el cambio de objetivo de "*to manage user privileges and attributes, and control the access to resources basing on policies*" a "*to manage user privileges and attributes, and control the access to resources basing on policies defined by the subject of care whom resources refer to*". Además en el punto de vista de la Empresa se debe incluir explícitamente el objeto de empresa *Subject of Care* que hereda del objeto *Person* y puede representar los roles *PolicyAdmin, PolicyRequester, Information Requester, Access Requester* y *Delegation Requester.* En el punto de vista de la Información no es necesario incluir un objeto de información para el SdA porque ya aparece en la normalización de HISA pero sí se debe establecer la correspondencia entre éste y el nuevo objeto de empresa definido.

Finalmente, el resto de requisitos que plantea el escenario POVO están completamente relacionados con la distribución de componentes como son por ejemplo la identificación unívoca de elementos, el descubrimiento de los mismos, las comunicaciones seguras, la autenticación a través de dominios, etc. Todos estos requisitos serán satisfechos por la plataforma tecnológica como será razonado en la especificación de las vistas dependientes de tecnología en los apartados siguientes.

6.3. El paradigma POVO: punto de vista de Ingeniería

En este apartado se describe la especificación del punto de vista de Ingeniería para la arquitectura de soporte de la POVO que tendrá correspondencias con los puntos de vista de la Empresa, la Información y Computacional establecidos en el apartado anterior. Una especificación de ingeniería se expresa mediante: una configuración de objetos de ingeniería (estructurados como clústeres, cápsulas y nodos), las actividades que ocurren dentro de esos objetos de ingeniería y las interacciones de esos objetos. Los objetos computacionales soportados por esta especificación de ingeniería son los presentados en las Figuras 4.18, 4.19 (control de acceso) y 4.29 (semántica). Estos objetos computacionales serán soportados por los objetos básicos de ingeniería correspondientes que estarán desplegados dentro de

clústeres en nodos, y por la infraestructura de ingeniería que utilizará nodos, núcleos, cápsulas, gestores de cápsulas, clústeres, gestores de clústeres y canales.

6.3.1. La infraestructura de ingeniería

En el punto de vista de Ingeniería se deben hacer concesiones a la tecnología debido a que es necesario formalizar la infraestructura que soporta las funciones relacionadas con la distribución. En nuestra propuesta la infraestructura de soporte se basa en tres modelos o estilos arquitecturales: las arquitecturas orientadas a servicios (SOA), las arquitecturas de servicios grid abiertos (OGSA*)* y su derivada semántica (*Semantic-OGSA,* S-OGSA). El estilo arquitectural SOA reparte la funcionalidad de la arquitectura entre los servicios que la componen los cuales pueden ser invocados y cuyas descripciones de interfaz pueden ser publicadas y descubiertas. OGSA particulariza este estilo arquitectural para el paradigma de computación en Grid añadiendo a los servicios la posibilidad de mantener información de estado entre invocaciones (aspecto que se había evitado en el estilo SOA). La iniciativa que trata de introducir capacidades semánticas en OGSA es S-OGSA que da un paso más en la sofisticación de la arquitectura de servicios grid. Cabe destacar que aunque se empieza a centrar el tema en una tecnología concreta (la computación en Grid) aún no se ha hecho una elección de la implementación específica que desplegaremos en el escenario real. Eso se hará en el punto de vista de Tecnología mientras que en este apartado nos limitaremos a utilizar los conceptos normalizados de los estilos arquitecturales comentados y que después podrán ser llevados a la práctica con diversas herramientas y soluciones comerciales.

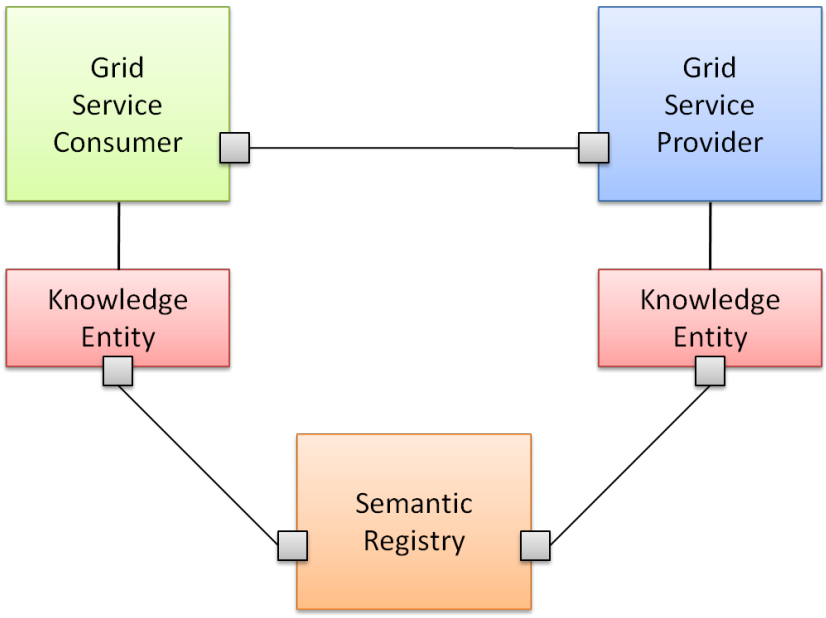

Figura 5.32. Diagrama abstracto de elementos y relaciones de la infraestructura de soporte

El esquema abstracto de relación entre servicios es el que se muestra en la Figura 5.32. En ella se observa que los servicios grid (*Grid Services*), y en general cualquier entidad de la Grid (servicios,

recursos, etc.), están asociados a entidades de conocimiento (*Knowledge Entities*) que los caracterizan y permiten su gestión a través de herramientas semánticas. Estas entidades de conocimiento o perfiles semánticos deben estar almacenadas y publicadas en un registro de servicios enriquecidos semánticamente (el registro semántico o *Semantic Registry*) que soporta la categorización de los mismos asistida por ontologías así como la búsqueda y descubrimiento de los perfiles semánticos registrados. Aunque en la figura se discrimina entre el consumidor (*consumer*) y el proveedor (*provider*), en general cualquier componente puede ser simultáneamente consumidor y proveedor de capacidades con respecto a otros componentes.

6.3.2. El sistema semántico de control de acceso

Una vez que tenemos el modelo abstracto que van a seguir los distintos servicios de la arquitectura, el siguiente paso consiste en especificar el punto de vista de Ingeniería a través de la configuración de objetos de ingeniería básicos, canales y resto de componentes. En este punto, en lugar de especificar todos los componentes de la arquitectura de soporte del escenario POVO, nos centraremos en el sistema semántico de control de acceso por tratarse de un componente con funcionalidad cerrada y porque es el elemento clave que será implementado en un sistema real para la validación de esta especificación. La formalización del punto de vista de Ingeniería va a constar de los siguientes elementos: diagramas de configuración de objetos básicos de ingeniería que soporten la funcionalidad de los objetos computacionales del sistema de control de acceso; diagramas con la configuración interna de los objetos de ingeniería más importantes y complejos; descripción de los canales y dominios de comunicación entre objetos; y finalmente, las correspondencias entre este punto de vista y los tres independientes de tecnología.

En primer lugar, en las Figuras 5.33 y 5.34 se muestra la configuración de objetos de ingeniería que soportan la funcionalidad del sistema de control de acceso. Ambas figuras forman parte del mismo diagrama que se ha separado en dos para facilitar su claridad. Así aunque tanto el elemento *ContextHandler* como el *SemanticRegistry2* aparecen en las dos figuras debe entenderse que son el nexo de unión entre las figuras y que sólo existe un objeto de cada tipo, complementándose ambas representaciones. Puede observarse asimismo cómo se mantiene el esquema de comunicación entre servicios impuesto por la infraestructura de distribución y basada en la publicación y descubrimiento de servicios. De este modo todos los servicios publican su funcionalidad en los *SemanticRegistry* y/o los utilizan para encontrar aquellos servicios más adecuados a sus necesidades. Aunque el diagrama presenta dos registros semánticos podría haber muchos más aunque siempre manteniendo de alguna forma la coherencia entre ellos. La distribución y desacoplamiento entre componentes se consigue a través de este estilo arquitectural. Cada vez que un servicio necesita la funcionalidad provista por otro consulta un registro semántico y utiliza la respuesta de éste para contactar con el servicio más adecuado. Previamente los servicios proveedores deben publicar sus características y perfiles semánticos en los registros para facilitar el descubrimiento.

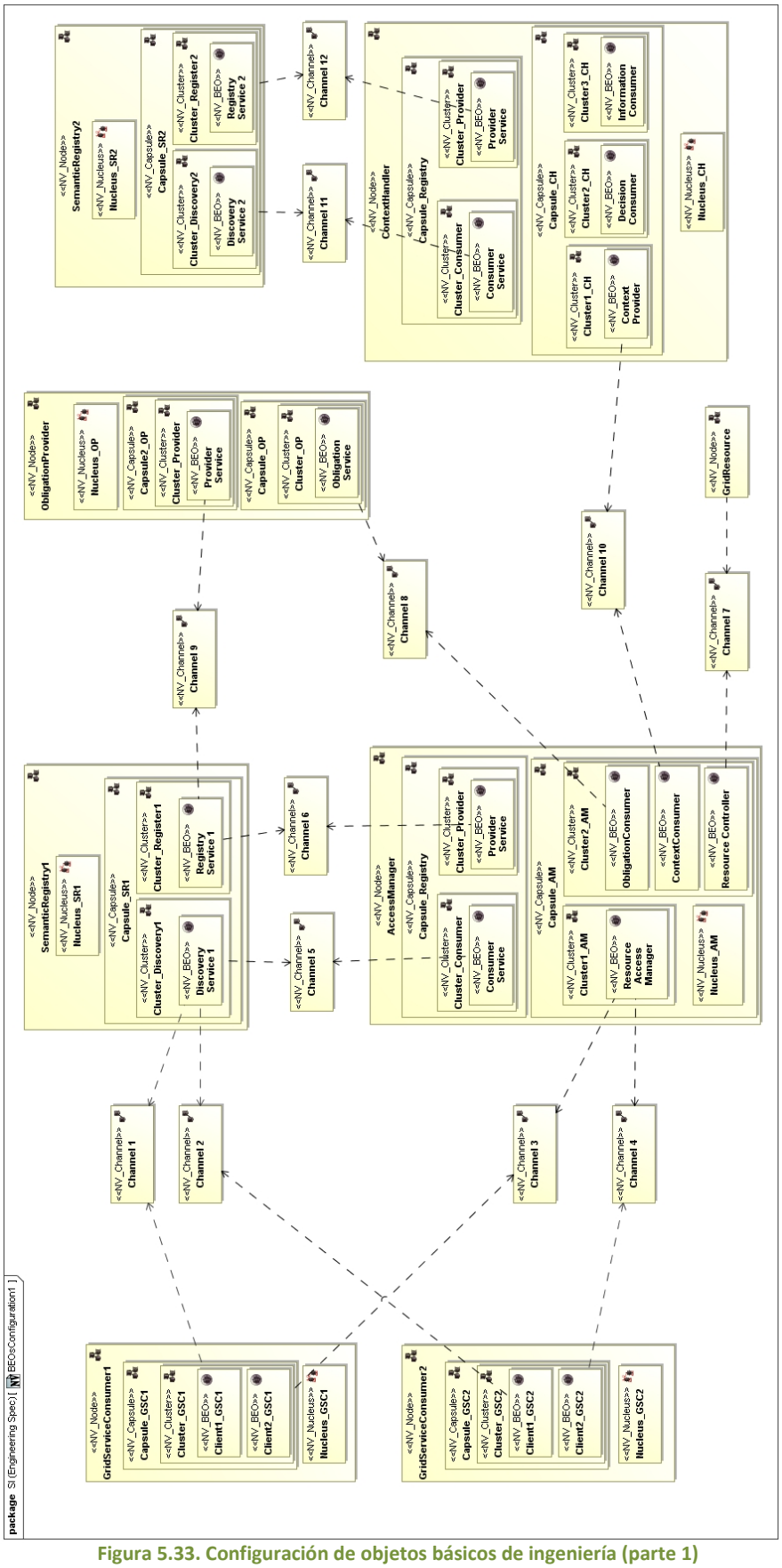

Figura 5.33. Configuración de objetos básicos de ingeniería (parte 1)

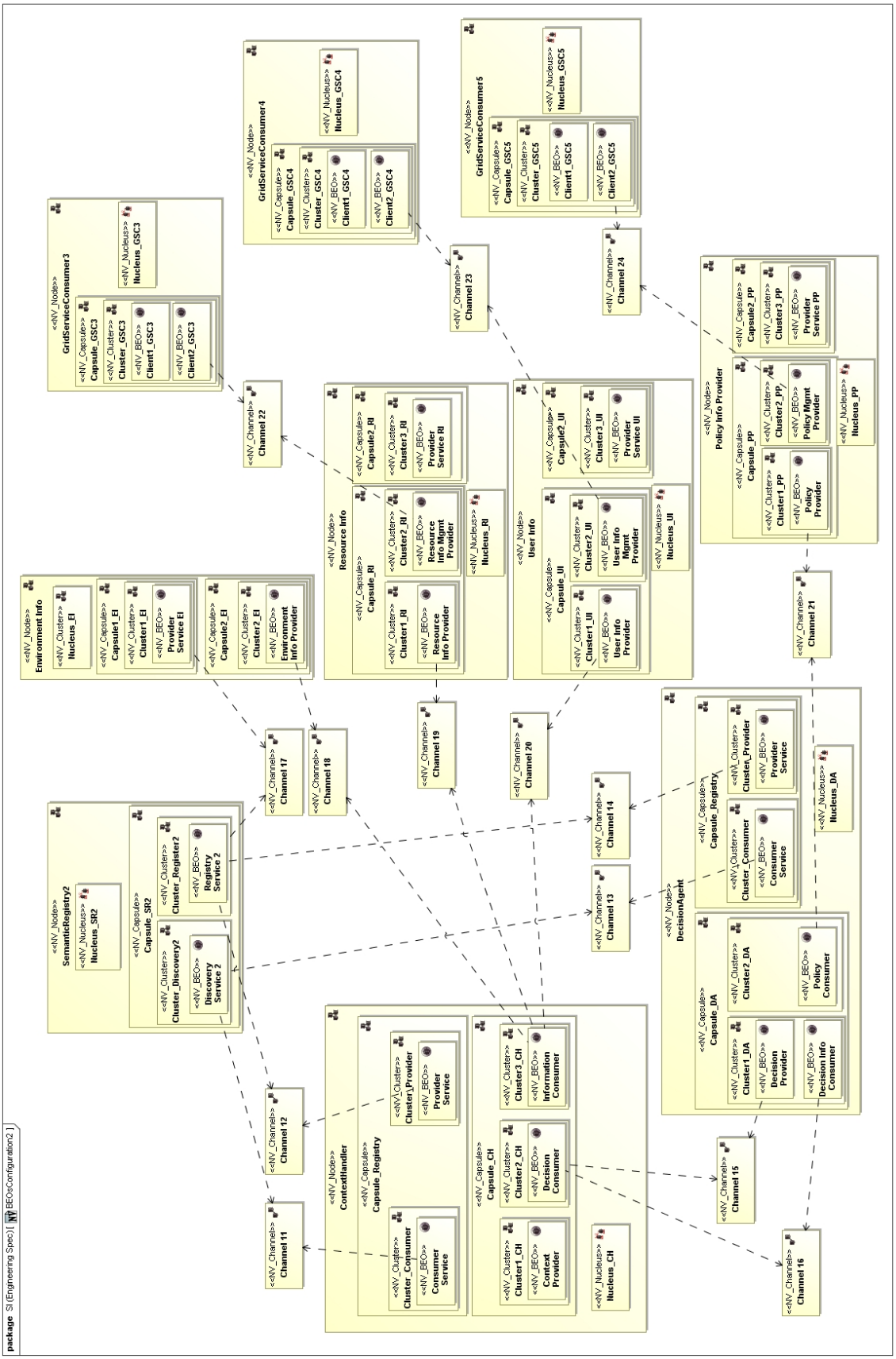

Figura 5.34. Configuración de objetos básicos de ingeniería (parte 2)

En este diagrama la única comunicación que no necesita pasar por un registro es la que se da entre un *AccessManager* y el recurso que protege ya que este enlace estará dedicado pues cada recurso debe estar asociado a un gestor de acceso. Algunos de los elementos (como los proveedores de información o los consumidores de servicios grid 3, 4 y 5) poseen los objetos básicos de ingeniería para comunicarse con el registro semántico correspondiente pero se ha simplificado el esquema por claridad.

Es necesario destacar que aunque en este diagrama se presenta un conjunto cerrado de elementos es sólo una muestra de lo que habría en un sistema real. Los elementos GridServiceConsumer hacen las veces de agentes de usuario actuando como usuarios del sistema completo y puede haber un número indeterminado. Igual ocurre con el conjunto de recursos (que varía dinámicamente) y al número de objetos que controlan el acceso a los mismos. También existirán proveedores de información distribuidos geográfica y administrativamente que podrán complementarse o replicar su información siempre manteniendo la coherencia de su contenido. Finalmente el escenario absolutamente descentralizado contempla numerosos agentes de decisión, gestores de contexto y servicios de obligaciones, evitando cualquier dependencia o cuello de botella con alguno de los elementos.

En las Figuras 5.35 a 5.38 se presentan las configuraciones internas de cuatro objetos de ingeniería: el *ContextHandler, AccessManager, DecisionAgent* y *Policy Info Provider.* Los cuatro poseen una cápsula que agrupa los objetos necesarios para la comunicación con un registro semántico si bien el *Policy Info Provider* no hace de cliente de ningún otro servicio y por tanto sólo presenta un canal de comunicación con el registro, el de proveedor. El resto de elementos tienen dos clústeres, uno que encierra el objeto básico de ingeniería para comunicarse con el registro como proveedor de servicio y otro como consumidor. El gestor de contexto (Figura 5.35) tiene tres clústeres con sendos objetos básicos de ingeniería. El *Context Provider* es enlazado por los gestores de acceso y es el que recibe la petición de acceso desde los mismos. Este objeto se comunica con el *Decision Consumer* que es el que interacciona con el agente de decisión que devolverá la autorización o denegación de acceso. Si es necesaria información adicional sobre el recurso, el solicitante de acceso o el entorno, el *Decision Consumer* pasa el control al *Information Consumer* que es el que se encarga de contactar con los diferentes proveedores de información sobre los usuarios, recursos o el entorno. La comunicación con el agente (o agentes) de decisión y los proveedores de información se hace de forma dinámica a través de la consulta del/los registro/s semánticos correspondientes. Por ello, aunque la formalización del gestor de contexto se haga en dos cápsulas, los procesos realizados internamente van pasando el control a los objetos de ambas cápsulas indistintamente en función de lo que se necesite.

El gestor de acceso (Figura 5.36) posee dos clústeres: uno contiene la funcionalidad de servicio proveedor, es decir, el objeto *Resource Access Manager* haciendo de enlace con los usuarios solicitantes de acceso a recursos; y el otro toma el papel de consumidor de otros servicios como son el *Context Handler* (a través del *Context Consumer*), el proveedor de obligaciones (que se enlaza mediante el objeto *Obligation Consumer*) y los recursos asociados (utilizando el objeto *Resource Controller*). Por su

parte el agente de decisión (Figura 5.37) separa su funcionalidad en dos clústeres. En el primero se pueden encontrar dos objetos que se relacionan con el/los *Context Handler/s*: el *Decision Provider* que recibe la petición de decisión y responde con la autorización o denegación de acceso y el *Decision Info Consumer* que permite solicitar al gestor de contexto información adicional en caso de que no se pueda tomar una decisión. El proveedor de decisión se comunica internamente con el consumidor de políticas (*Policy Consumer*) que es el encargado de solicitar al proveedor de políticas aquellas aplicables en el acceso que se esté evaluando. Análogamente a como se hace en el resto de elementos, la comunicación con el (o los) proveedor de políticas se hace de forma dinámica a través de la consulta a un registro semántico y el descubrimiento del servicio adecuado. Por último, como ejemplo de proveedor de información tenemos el *Policy Info Provider* (Figura 5.38) aunque el resto de proveedores de información (de usuarios, recursos y entorno) están configurados de manera análoga. Este tipo de elementos siempre actúan como proveedores de capacidades por eso que en la cápsula de registro no aparezca el clúster ni el objeto de ingeniería dedicados para el rol de consumidor. El *Policy Info Provider* proporciona dos tipos de capacidades: da servicio a los agentes de decisión a través del objeto *Policy Provider* y permite la gestión de las políticas por un cliente o administrador lo que se realiza mediante el objeto *Policy Mgmt Provider.*

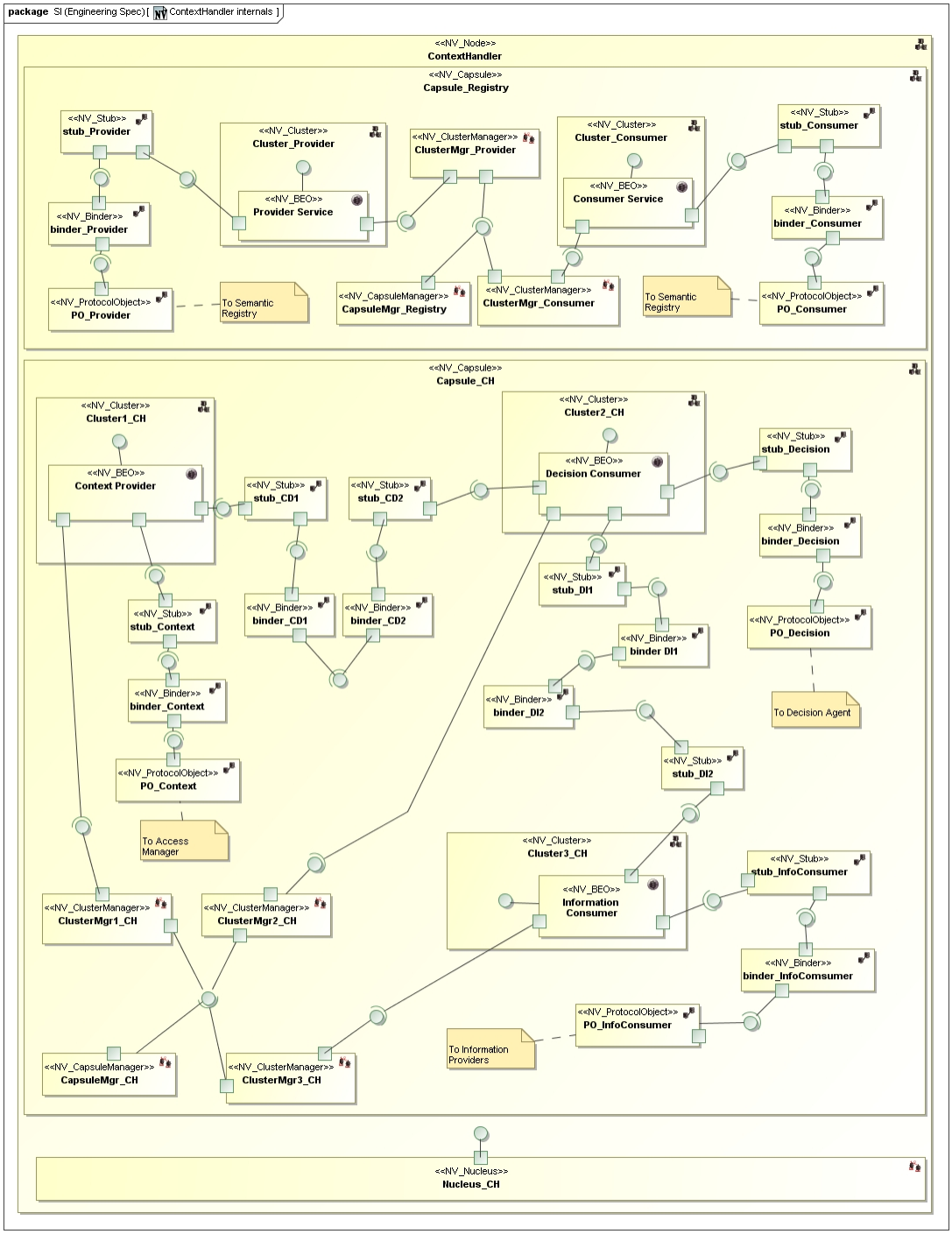

Figura 5.35. Configuración interna del Context Handler

Como se puede ver en las Figuras 5.33 y 5.34 existen numerosos canales entre los distintos objetos de ingeniería. El concepto de canal en el punto de vista de Ingeniería es definido como una configuración de *stubs*, *binders* y *protocol objects* que proporciona una vinculación entre un conjunto de interfaces a objetos básicos de ingeniería, y a través de la cual pueden darse interacciones. Un *stub* es un objeto de ingeniería que interpreta las interacciones que se llevan a cabo en el canal, y realiza cualquier transformación o monitorización necesaria basada en esa interpretación. Por su parte, un *binder* es el objeto de ingeniería que permite mantener la vinculación entre objetos básicos de ingeniería.

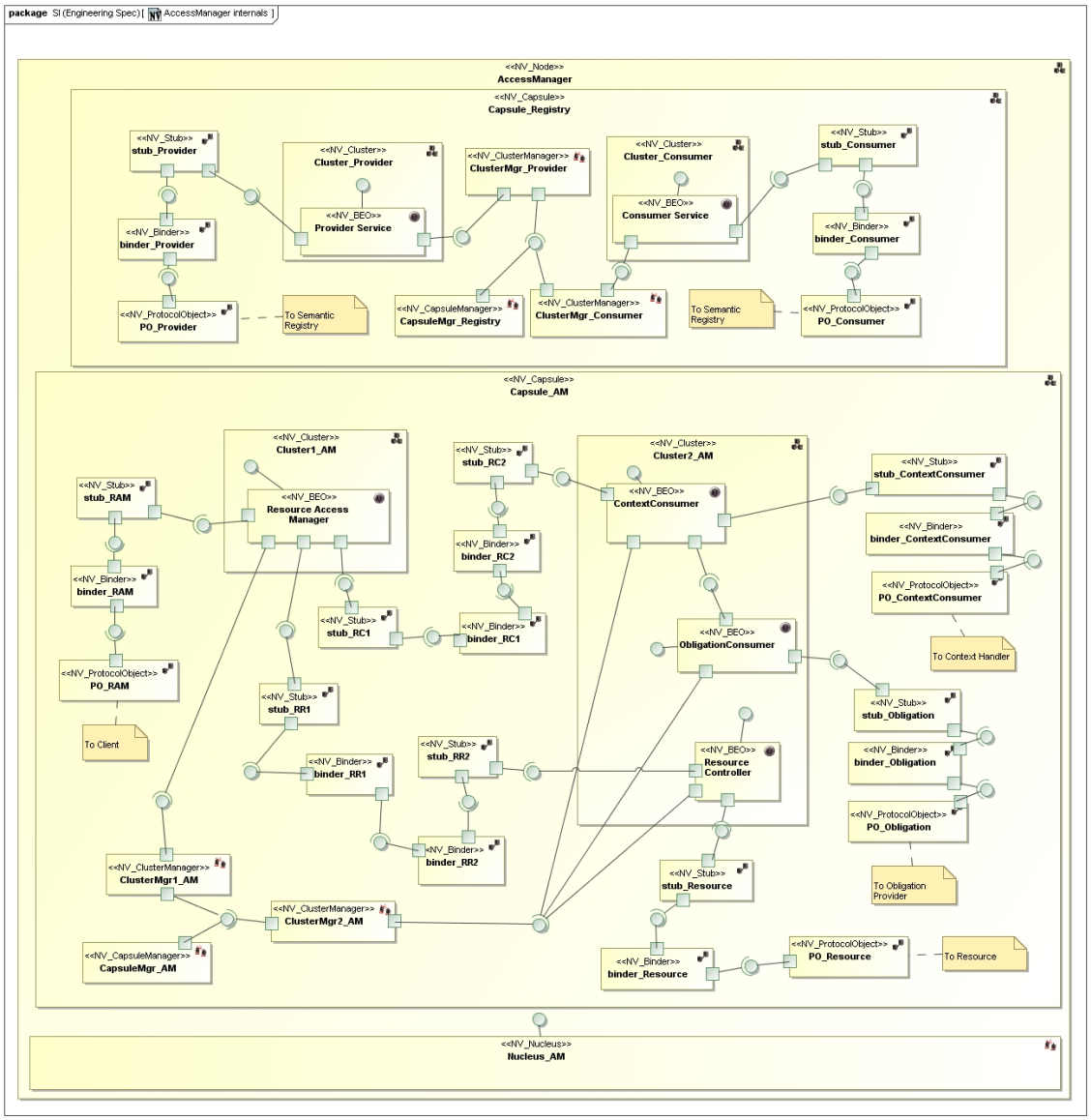

Figura 5.36. Configuración interna del Access Manager

Finalmente, el objeto protocolo (*protocol object*) es el objeto de ingeniería que comunica con otro objeto protocolo en el mismo canal para lograr la interacción entre objetos básicos de ingeniería (que pueden estar en diferentes clústeres, cápsulas o nodos). En la Figura 5.39 se muestra la configuración interna general de los canales. Asimilando estos conceptos a la tecnología de Servicios Web, el *stub* y *binder* del consumidor del servicio son el *stub* del cliente web, y los del proveedor equivalen al *skeleton* del servidor web.

Los dominios de comunicación (*communication domains*) agrupan aquellos objetos de protocolo capaces de interactuar entre sí y permitir la vinculación de sus respectivos objetos básicos de ingeniería. En la Figura 5.40 se han especificado dos dominios de comunicación principales: SAML/XACML y HTTP.

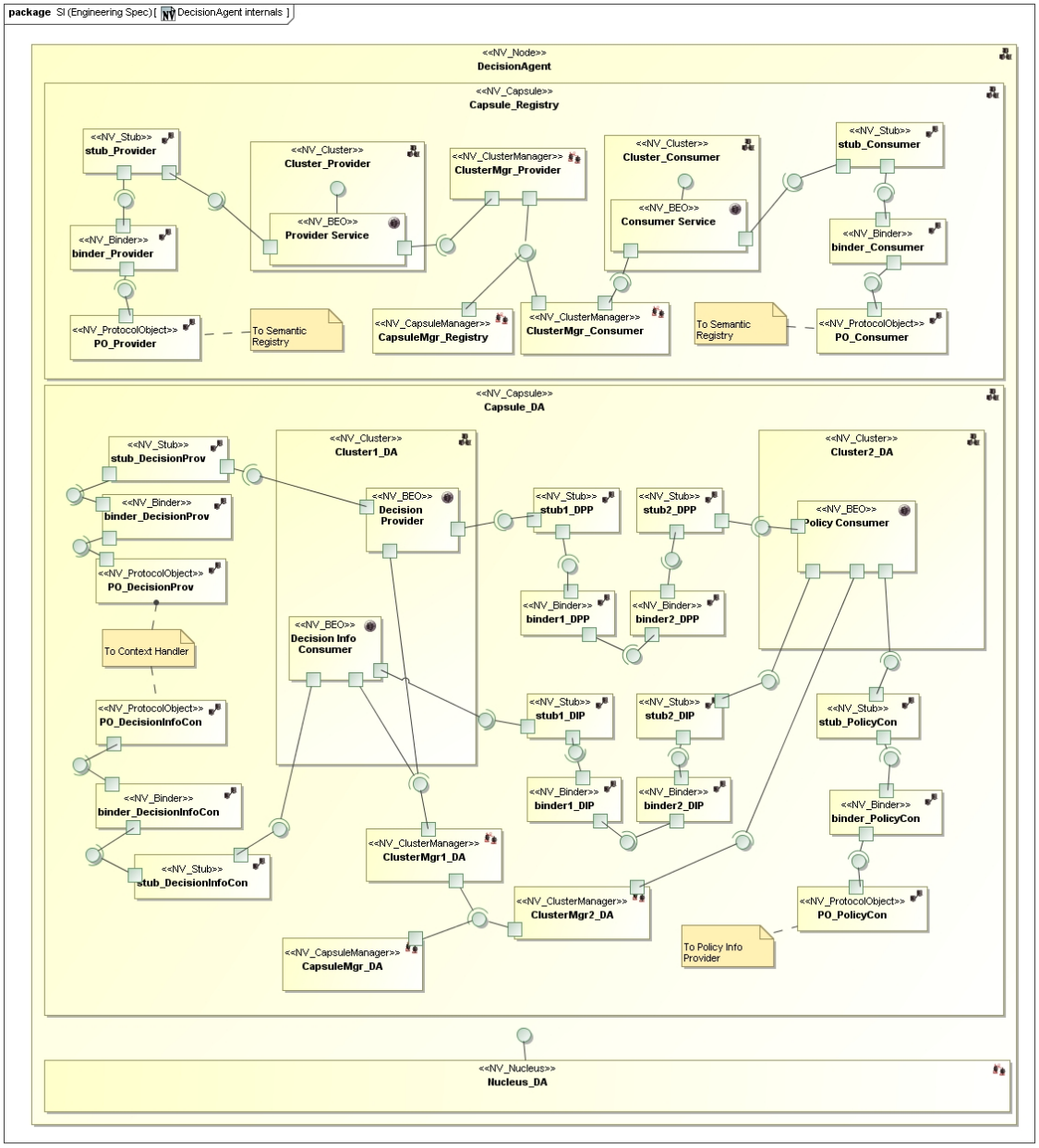

Figura 5.37. Configuración interna del Decision Agent

El primero corresponde a las comunicaciones que se dan entre todos los objetos directamente implicados en el control de acceso a los recursos. Buscando la normalización del sistema todas estas comunicaciones se hacen en base al protocolo SAML y el lenguaje XACML como se especificará en el punto de vista de Tecnología. Las comunicaciones del sistema de control de acceso con los elementos externos (es decir, con los recursos, clientes y administradores) se hará a través del protocolo de uso general HTTP y SOAP como corresponde a los servicios web.

Finalmente, el punto de vista de Ingeniería contempla la especificación de las funciones que permiten la distribución del sistema. El marco de trabajo ODP y la especificación OGSA identifican un conjunto básico de funciones necesarias. En la Tabla 5.II se presentan estas funciones y la correspondencia entre

las de RM-ODP y OGSA. En caso de que la función esté identificada en ambas propuestas se mantendrá la nomenclatura de RM-ODP por tratarse del esfuerzo estándar.

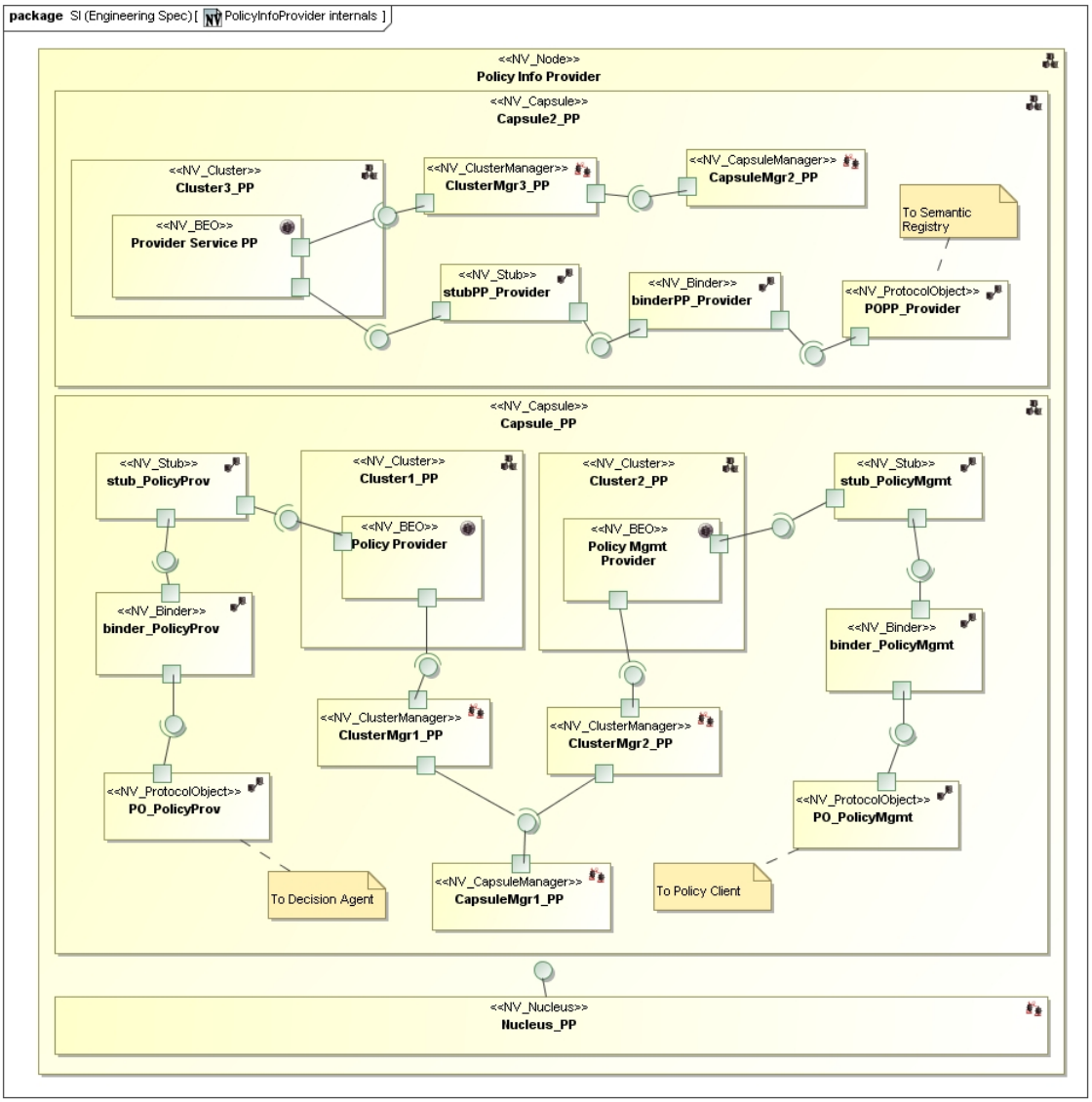

Figura 5.38. Configuración interna del Policy Info Provider

6.3.3. Correspondencias del punto de vista de Ingeniería

Las correspondencias entre los puntos de vista de la Empresa y de Ingeniería se pueden separar en dos grupos: uno para las relaciones entre objetos de empresa y objetos de ingeniería y otro para las relaciones entre procesos de empresa y canales de ingeniería implicados. Por otro lado, RM-ODP declara que no existen correspondencias relevantes entre los puntos de vista de la Información y el de Ingeniería, por tanto, para terminar la especificación del punto de vista de Ingeniería basta con identificar las relaciones que existen entre sus objetos y los del punto de vista Computacional. Básicamente se trata de las relaciones entre los objetos computacionales de alto nivel y los de ingeniería.

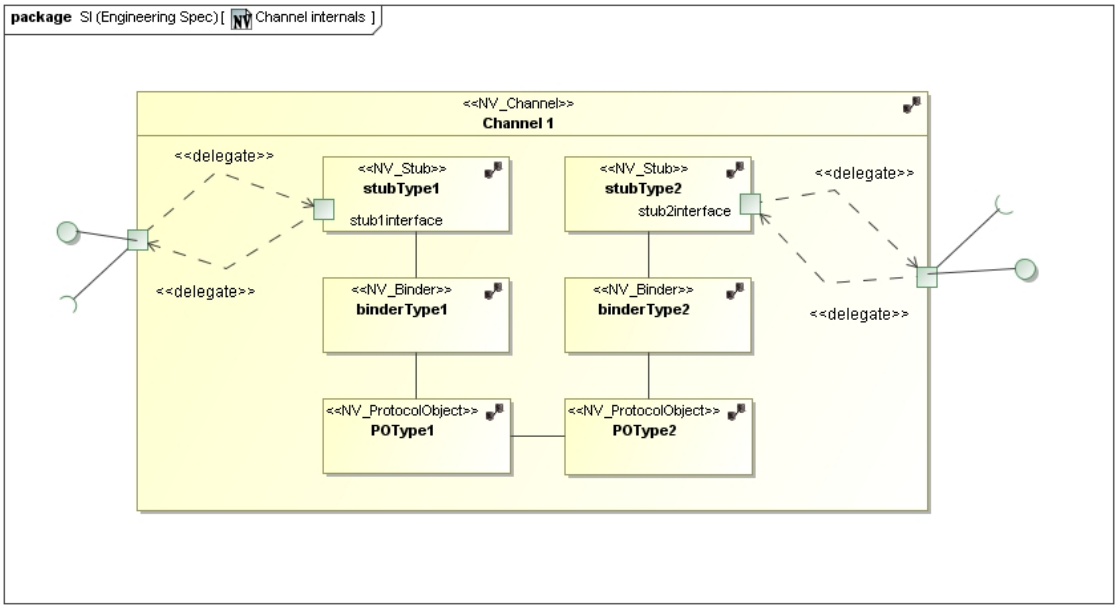

Figura 5.39. Configuración interna de un canal entre objetos básicos de ingeniería

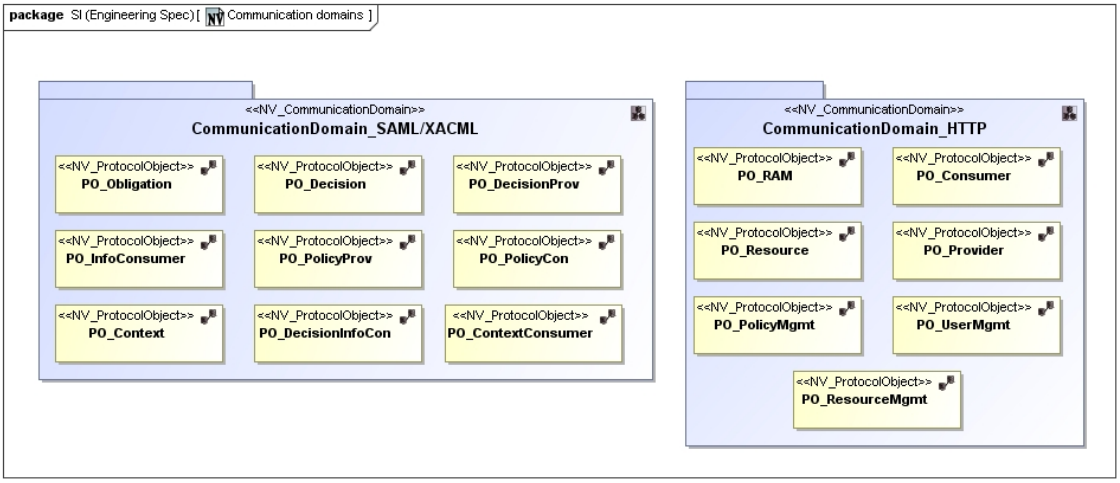

Figura 5.40. Muestra de los dominios de comunicación implicados en el control de acceso

6.4. El paradigma POVO: punto de vista de Tecnología

El punto de vista de Tecnología cubre los aspectos de implementación del sistema ODP identificando los elementos hardware, software y de red del mismo. Debido a que el sistema presentado puede ser desplegado en un número indeterminado de escenarios gracias a la flexibilidad que aporta la distribución de los componentes, la formalización de este punto de vista se resume con diagramas de alto nivel que muestran aspectos comunes a todos los posibles escenarios. Aparte de la formalización propiamente dicha según las directrices de RM-ODP y UML4ODP, se incluyen en este apartado las implementaciones de todos los componentes que componen el sistema de control de acceso así como un razonamiento sobre el funcionamiento del mecanismo presentado. Para ello se presenta: la

ontología POVO federada y todas sus sub-ontologías, las políticas de control de acceso (cómo son implementadas y cuál es el proceso que se sigue para la toma de decisiones) y las implementaciones de todos los componentes del mecanismo haciendo especial hincapié en cómo se han adoptado los distintos estándares de comunicaciones y del dominio sanitario para construir una solución interoperable y abierta.

Funciones RM-ODP		Funciones OGSA
Funciones de gestión	Gestión de nodos	---
	Gestión de objetos	---
	Gestión de clústeres	---
	Gestión de cápsulas	---
Funciones de coordinación	Notificación de eventos	Notificaciones
	Punto de comprobación y recuperación	---
	Desactivación y reactivación	---
	Función de grupo	Orquestación
	Función de replicación	---
	Función de migración	---
	Rastreo de referencia de interfaz	Descubrimiento
	Función de transacción	Transacciones
	Función de transacción ACID	---
Funciones de repositorio	Almacenamiento	Gestión de almacenamiento
	Organización de información	---
	Función de reubicación	Gestión de localización
	Repositorio de tipos	Catálogos metadatos
	Función de intermediación	Descubrimiento
Funciones de seguridad	Control de acceso	Autorización
	Auditoría de seguridad	Auditoría y registro seguro
	Autenticación	Autenticación
	Integridad	---
	Confidencialidad	---
	No-repudio	---
	Gestión de claves	---

Tabla 5.II. Conjunto de funciones de RM-ODP y OGSA que soportan la distribución

Siguiendo la elección del procesamiento en Grid como plataforma tecnológica de soporte se ha optado por utilizar *The Globus Toolkit* [65] como software de intermediación. Este conjunto de herramientas, actualmente en la versión 5, proporcionan capacidades de bajo nivel para el soporte de Grids computacionales conforme a la especificación OGSA. No se entra en mayor detalle en el uso de *The Globus Toolkit* más que para indicar que permitirá que los diferentes componentes actúen como servicios grid y sean capaces de heredar funcionalidades relacionadas con la distribución de elementos de la plataforma (migración, descubrimiento, almacenamiento de datos, virtualización de recursos, etc.).

6.4.1. La ontología POVO federada

Para alcanzar el grado de variabilidad requerido por las políticas de control de acceso y potenciar a su vez las capacidades de gestión semántica de los diferentes componentes de la arquitectura se ha diseñado y desarrollado una ontología de los conceptos implicados en el escenario POVO. Con respecto al sistema de control de acceso, utilizando esta ontología el SdA puede tener un control versátil sobre el acceso a sus recursos determinando: los actores potenciales que pueden acceder a ellos, la naturaleza de la información o las enfermedades relacionadas con él, fechas de creación y publicación, autores, ubicación física de los recursos, etc. La ontología propuesta cubre los aspectos básicos del escenario POVO y está compuesta de: una ontología de elementos del dominio sanitario; otra centrada en los descriptores de los recursos; una tercera cubriendo los aspectos de seguridad y control de acceso; y una última basada en la infraestructura del modelo POVO, en este caso particularizada para tecnologías Grid.

La ontología de elementos sanitarios (Figura 5.41) cubre los actores involucrados en el dominio e identifica tres grupos principales: personas, organizaciones y dispositivos sanitarios. El grupo *Person* incluye todos los individuos reales desde el propio SdA y los profesionales sanitarios (médicos, enfermeros…) a sus parientes, amigos o cuidadores. Cada individuo tiene un atributo que identifica su relación con el SdA y es el que se usa para alimentar el mecanismo de control de acceso. Algunas relaciones existentes por defecto en el sistema son: PRIM (de *Primary attention healthcare profesional*) para designar el médico de familia del SdA, BBY (de *Baby*) para sus hijos, OCR (*Other carer*) para cuidadores no profesionales y FRE (*Friend*) para los amigos. Además de las incluidas por defecto, el SdA podrá crear nuevas relaciones que se ajusten a sus necesidades en cada momento. El segundo grupo de actores cubre las organizaciones e incluye las de tipo sanitario por un lado (hospitales, departamentos, clínicas, farmacias, etc.) y otras entidades que pueden jugar un rol en la asistencia sanitaria del SdA. Se pueden encontrar numerosos tipos de organizaciones implicadas en el mantenimiento de la salud y el bienestar como gimnasios o centros dietéticos, compañías de seguros, laboratorios, grupos de investigación independientes, entidades que despliegan y mantienen recursos sanitarios, etc. Finalmente el tercer grupo de actores incluye los dispositivos sanitarios que tienen un rol importante en el entorno sanitario ya sea como fuentes o sumideros de información e incluso algunas veces pudiendo realizar tareas en representación de personas, ya sean profesionales sanitarios o el propio SdA.

Por otro lado y dentro de la misma ontología, se han tenido en cuenta dos categorías de objetos cuyo acceso debe estar controlado: la información y los recursos. Ambos grupos tienen atributos que los identifican además de permitir el control de versiones y auditoría. La información está estructurada en elementos que cubren todos los aspectos relacionados con la salud como resultados de pruebas diagnósticas, medicaciones, rutinas de ejercicios, hábitos alimenticios, imágenes, información genómica, etc. Los recursos (bases de datos, herramientas software de modelado y simulación, etc.) son desplegados y mantenidos por las organizaciones y utilizan información sobre el SdA. Esto es capital para determinar el control de acceso a esos recursos ya que existen dos tipos de políticas implicadas: las determinadas por los propietarios reales de los recursos (las organizaciones) y las que impone el SdA

sobre su información que es utilizada en las operaciones de estos recursos. Una discusión a este respecto se desarrolla más adelante. Finalmente, esta ontología recoge los tipos genéricos de actividades del dominio sanitario como son las que realizan los profesionales (*Healthcare_Provider_Activity*), las automatizadas por dispositivos (*Automated_Healthcare_Activity*) o las que realizan los propios SdAs (*Health_Selfcare_Activity*).

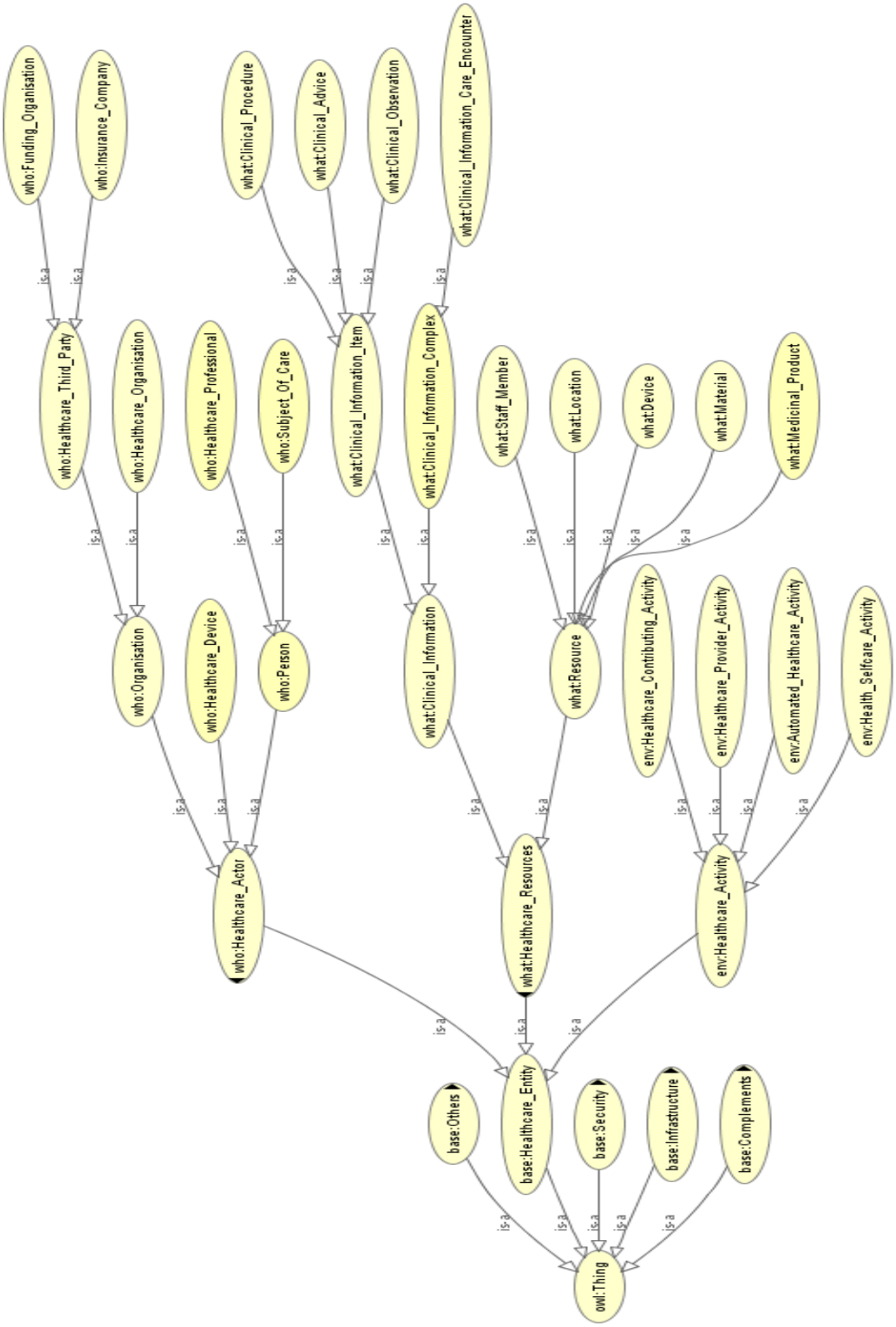

Figura 5.41. Ontología POVO – Parte I: Ontología de entidades sanitarias (Healthcare_Entity)

La información será categorizada mediante descriptores (Figuras 5.42 y 5.43) que permitirán indicar su naturaleza (datos demográficos, estilo de vida saludable, hábitos alimenticios, enfermedades, etc.) y si tiene carácter anónimo o una persona (ya sea el SdA u otra) puede ser relacionada con esa información. Además de esta clasificación, que puede ser automatizada en mayor o menor medida, se han incluido otros descriptores que pueden ser utilizados por el SdA y que le facilitan la creación de políticas de acceso. Por ejemplo, el SdA puede determinar qué información está disponible para consulta y uso y cuál es exclusiva para su asistencia; establecer un esquema de control basado en niveles de confidencialidad; y un amplio rango de posibilidades. Otros aspectos han sido introducidos para reflejar la relación del SdA con las organizaciones (contactos, encuentros, eventos y periodos de asistencia, etc.) y para la clasificación de actividades considerando quién las ejecuta (automatizadas por los dispositivos o realizadas por cuidadores, profesionales o el propio SdA). La muestra de atributos presentada en las Figuras 5.42 y 5.43 pretende ser un esbozo de los aspectos que podrían ser cubiertos por la ontología completa y en ejecución en el escenario real. El resultado es fruto de un esfuerzo de análisis y conciliación de varias referencias normativas. Por ejemplo, el atributo *Purpose_of_Use* está basado en su homónimo del perfil de SAML para entornos sanitarios (XSPA SAML [141]) y se ha mantenido la categorización de roles en estructurados y funcionales como indica la norma ISO21298 [155]. Los atributos de contexto *Locality* y *Time* han sido extraídos de las normas XSPA SAML y XACML, respectivamente. Paralelamente en esta ontología de complementos se encuentran también los descriptores de contexto que indican situaciones anómalas que requieren la ejecución de tareas extraordinarias. Algunos casos son los escenarios de emergencia o "*break-glass*" en los que la salud pública o del SdA están en riesgo y se requieren medidas aun sin el consentimiento del SdA. Otros complementos tienen en cuenta aspectos temporales para facilitar la gestión semántica del tiempo.

La tercera ontología que se ha diseñado (Figura 5.44) cubre los aspectos de infraestructura y tecnologías teniendo una alta caracterización de procesamiento en Grid, identificando los conceptos y elementos principales de una plataforma de este tipo. Por otro lado, dada la orientación a servicios que se ha intentado plasmar en el sistema, existe un conjunto de perfiles de acceso a los mismos (*AccessProfile*). Estos perfiles no están relacionados con la disciplina de seguridad sino con el mecanismo que utilizan los servicios para publicar sus capacidades y ser descubiertos y utilizados por otros. En la medida que se utilicen otras plataformas tecnológicas esta ontología tendría que ser extendida con los conceptos específicos de cada una de ellas. La cuarta ontología tiene en cuenta los aspectos de seguridad de la POVO (Figura 5.45). Se ha profundizado en la disciplina de autorización aunque el resto de aspectos de seguridad deberían ser incluidos en esta ontología (de forma similar a como aparece la auditoría). En cuanto al mecanismo de control de acceso se especifican los usuarios y recursos que a su vez están enlazados con los actores y recursos de la ontología de componentes sanitarios. Se ha utilizado el conjunto de acciones definido en la norma XSPA SAML para completar las acciones que pueden realizar los actores sobre recursos de información.

Finalmente, se debe aclarar que la ontología desarrollada no pretende convertirse en una solución completa sino que sólo representa los conceptos básicos de una POVO ilustrando cómo podría resolverse el control de acceso. A pesar de eso es una solución flexible y escalable que puede, a través de los mecanismos de importación de ontologías, alimentarse de otras iniciativas ampliamente aceptadas ya sean de terminologías clínicas (SNOMED [108], GALEN [206]…), de pruebas y resultados de laboratorio (LOINC [109]) u otras. Este aspecto se ha querido representar a través de la inclusión de la ontología *Basic Formal Ontology* (BFO) [207] (Figura 5.45) que es la meta-ontología a partir de la cual se están especificando todas las ontologías de la iniciativa *The Open Biological and Biomedical Ontologies Foundry* (OBO Foundry) [208] la cual busca desarrollar con rigor un conjunto ampliable y extensible de ontologías en el dominio biológico a través de una metodología de composición de pequeñas ontologías.

Figura 5.42. Ontología POVO – Parte II: Ontología de complementos y atributos (Complements I)

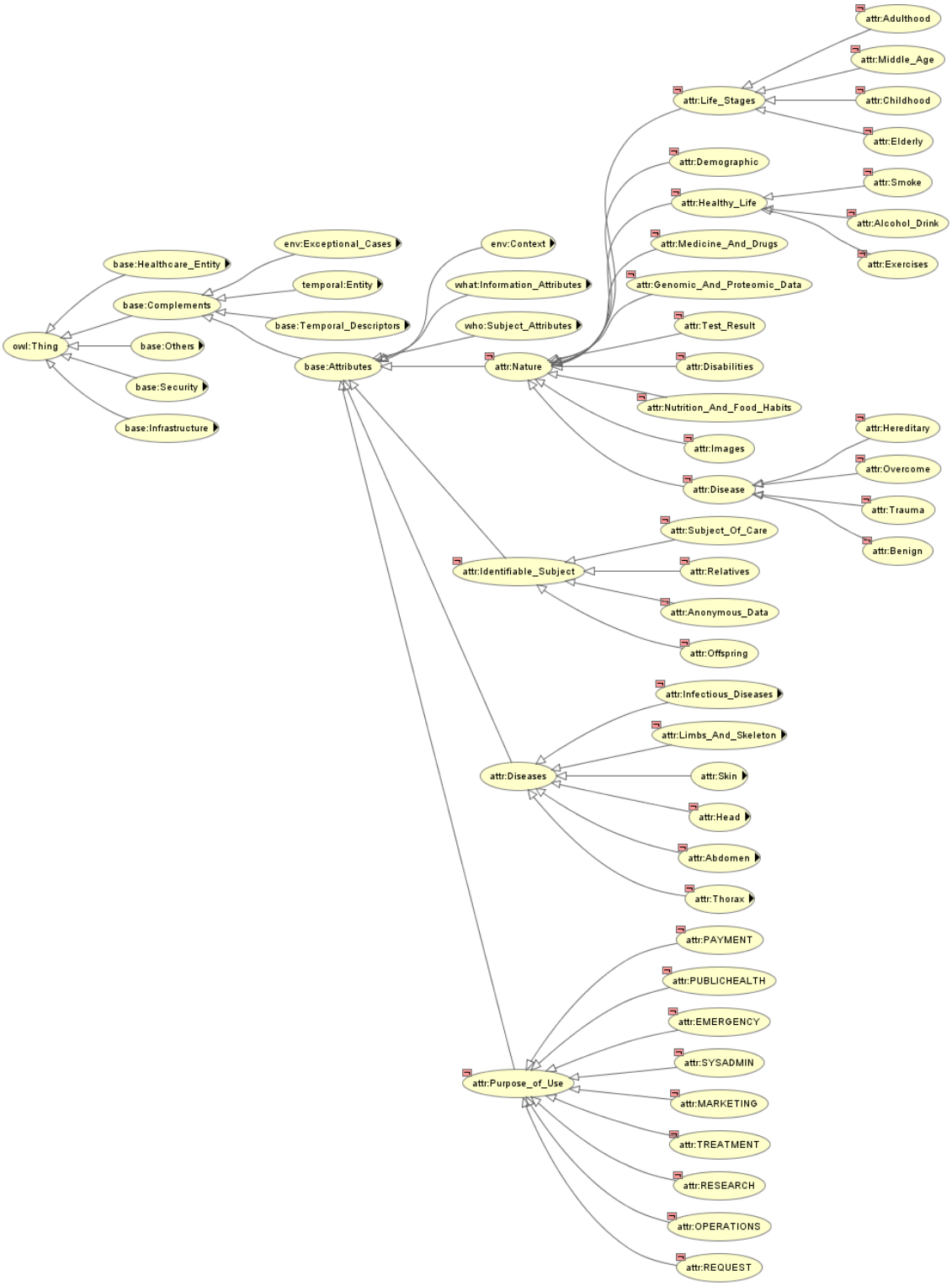

Figura 5.43. Ontología POVO – Parte III: Ontología de complementos y atributos (Complements II)

Figura 5.44. Ontología POVO – Parte IV: Ontología de infraestructura (Infrastructure)

6.4.2. Las políticas de control de acceso

Las políticas que gobiernan el acceso a los recursos se definen mediante reglas de lenguaje SWRL y están basadas en los conceptos de las ontologías soportando el escenario POVO. En esta propuesta una política es una regla de tipo "Horn-like" en la que el antecedente está compuesto de elementos (actores, recursos, atributos, características del entorno…) que condicionan la decisión y el consecuente especifica si la petición solicitada está permitida o prohibida. Dos ejemplos de políticas de control de acceso y su especificación como reglas SWRL son:

1. Política general o de grano grueso - "permitir el acceso a los recursos de mi POVO a todos los profesionales sanitarios"

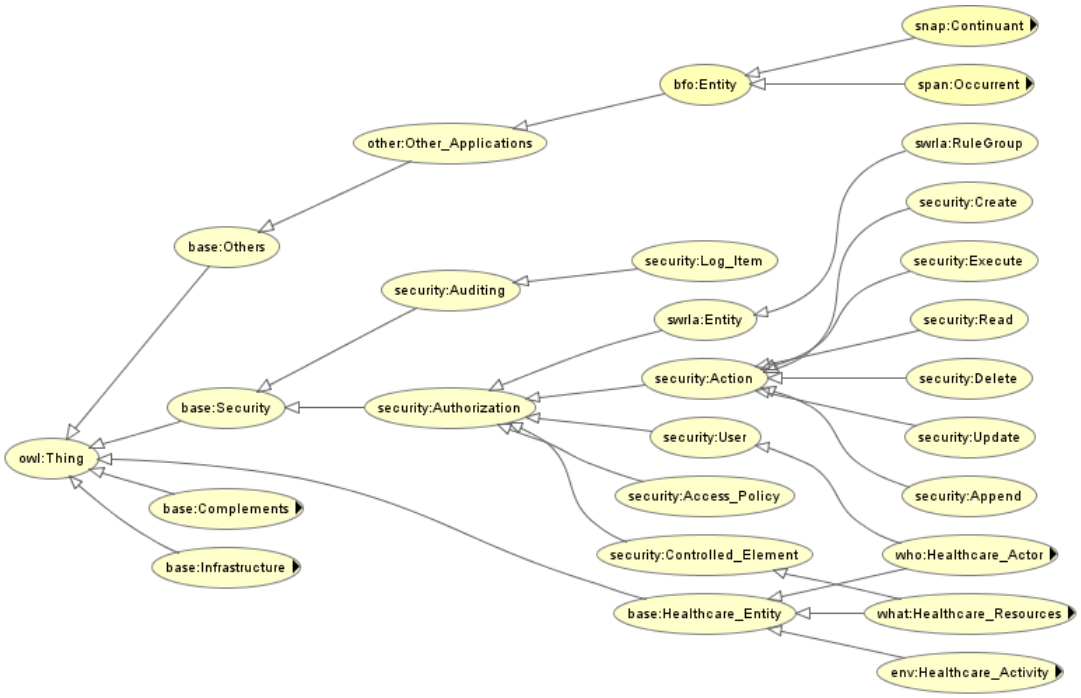

Figura 5.45. Ontología POVO – Parte V: Ontología de seguridad (Security) y otros dominios (Others)

2. Política particular o de grano fino – "permitir a mi esposa ver toda mi información relacionada con enfermedades de transmisión sexual desde el año 2000 y en el que terceras personas no estén involucradas"

who:Person(?per) ^ who:hasRelation(?per, who:SPOUSE) ^ what:Clinical_Information(?inf) ^ attr:Sexual_Organs(?dis) ^ isRelatedTo(?inf,?dis) ^ attr:Subject_Of_care(?soc) ^ isRelatedTo(?inf,?soc) ^ what_creationTime(?inf,?time) ^ temporal:notBefore(?time,"2000-1-1")

↓

actionPermitted(?per,?inf)

La interpretación de una regla como las anteriores es: si las condiciones especificadas en el antecedente son verdaderas (es decir, si existen individuos OWL que satisfacen todas las cláusulas), entonces la propiedad *actionPermitted* (o *actionProhibited*) debe crearse entre el/los actor/es y el/los recurso/s implicados. Este proceso de comprobación de reglas y creación de propiedades será realizado por el motor de inferencia del agente de decisión como se explica más adelante.

En este punto cabe señalar cómo se satisfacen los requisitos identificados en el escenario POVO para las políticas de control de acceso. De acuerdo con los requisitos 2, 3 y 8 se tienen tres entidades capaces de crear políticas de control de acceso: el SdA que ostenta la autoridad sobre su información y puede definir reglas conforme a sus deseos; los proveedores de los recursos incluidos en la POVO que son los que controlan el acceso a ellos pero no pueden interferir con las políticas definidas por el SdA sobre su

información; y finalmente los organismos legislativos regionales e internacionales que especifican políticas que se aplican en escenarios excepcionales en los cuales la autoridad del SdA puede ser invalidada. Las especificaciones de control de acceso impuestas por estos tres grupos deben coexistir y no pueden aparecer incoherencias que resulten en decisiones de acceso erróneas. La resolución metodológica de este trabajo está basada en los siguientes puntos:

- Las políticas establecidas por los organismos legislativos tienen el nivel más alto de prioridad. Así una decisión impuesta por una política de este grupo anula cualquier política aportada por el usuario o los proveedores de recursos. Entre las políticas legislativas se pueden encontrar como ejemplo las que permiten al SdA acceder a toda su información, las que consideran escenarios de emergencias (con riesgo para la salud del SdA, enfermedades de declaración obligatoria, riesgos de salud pública, etc.) y otros casos como el de menores o incapacitados. Debe ser remarcado que este grupo de políticas sólo actúa en casos excepcionales mientras que en la ejecución normal del sistema la autoridad del SdA rara vez será anulada.

- En un escenario normal (es decir, sin políticas legislativas activas) las políticas definidas por el SdA son responsables del control de acceso. Los ejemplos mostrados anteriormente ilustran el tipo de políticas incluidas en este grupo.

- Un tercer grupo está formado por las políticas que establecen los gestores de los recursos. Éstas pueden regular el acceso a los recursos pero nunca a la información de los SdAs. Como en general los recursos utilizarán información para realizar sus tareas, una política que rija el acceso a un recurso debe tener en cuenta las políticas de control de acceso a la información utilizada por el recurso y definidas por el SdA. Así puede suceder que un usuario tenga acceso a un recurso (porque las políticas del gestor correspondiente lo permiten) pero no a la información utilizada por el mismo (porque así lo ha determinado el SdA). Esta situación puede darse a la inversa, un usuario puede tener acceso a la información de un SdA pero no al uso de un recurso gestionado por, por ejemplo, una organización sanitaria privada.

El proceso de toma de decisiones realizado por el agente de decisión sigue el esquema mostrado en la Figura 5.46. Cuando este componente recibe toda la información necesaria a través del gestor de contexto, primero combina la ontología de la POVO con las reglas SWRL relativas al grupo legislativo, y el razonador ejecuta la inferencia. Los axiomas inferidos son incorporados a la ontología y se utiliza el lenguaje de consultas SQWRL para verificar la existencia de la propiedad *actionPermitted* entre el solicitante del acceso y el recurso solicitado. Si las políticas legislativas han especificado esta propiedad, entonces se permite el acceso. En otro caso, se realiza una nueva consulta para encontrar la propiedad *actionProhibited* y si existe se deniega el acceso al solicitante. En caso de que en este punto no se pueda tomar una decisión (esto significa que el acceso no está sometido a las reglas legislativas), se ejecuta una vez más el proceso de inferencia pero esta vez con todas las reglas, es decir, añadiendo las políticas

definidas por el SdA y por los propietarios de los recursos. De nuevo se realizan dos consultas SQWRL para verificar la existencia de propiedades de permiso o prohibición. En este punto, los escenarios posibles son:

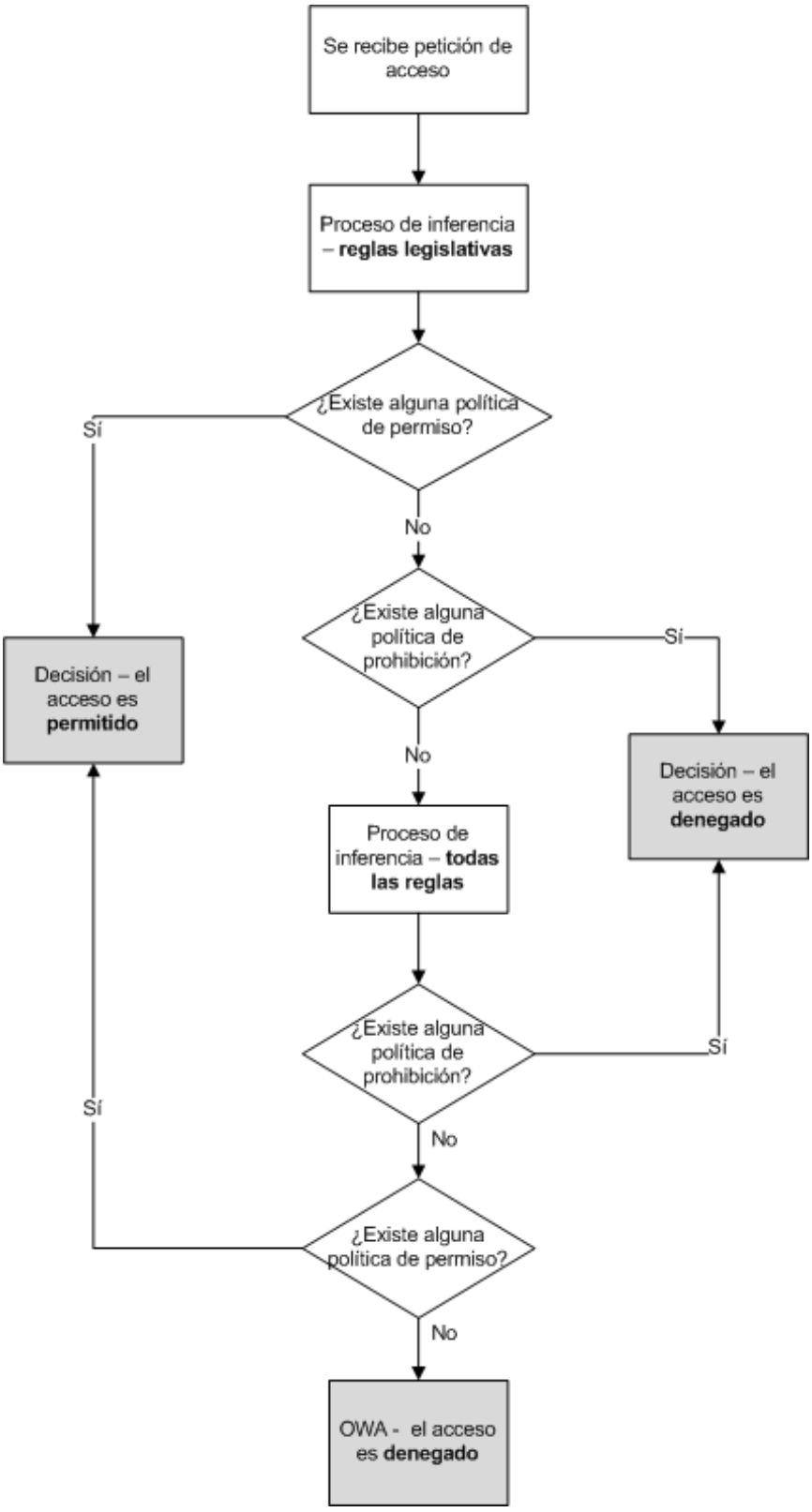

Figura 5.46. Esquema del proceso de toma de decisiones realizado por el motor de inferencia

1. Hay una propiedad de permiso (o prohibición) entre el usuario y el recurso, por tanto se toma la decisión de aceptación (o denegación) de acceso.

2. Dos o más políticas definidas por el SdA son incoherentes y existe tanto la propiedad de permiso como la de prohibición al mismo tiempo. En este caso se toma la decisión más conservativa denegando el acceso.

3. No hay ninguna política controlando el acceso solicitado y, como el lenguaje está basado en OWA, no se puede tomar ninguna decisión. Se resuelve este escenario haciendo que el agente de decisión deniegue el acceso y lo comunique al SdA para que éste especifique (si es su deseo) la regla que controlará este acceso en el futuro.

6.4.3. La implementación de los componentes del mecanismo de control de acceso

En este apartado se describen las implementaciones de los diferentes servicios que forman parte del mecanismo de control de acceso, es decir, los gestores de acceso a los recursos, los gestores de contexto, el proveedor de políticas, los proveedores de información, el agente de decisión y el servicio de obligaciones. Para la comunicación entre componentes se han adoptado dos de los estándares más extendidos para la comunicación de consultas y decisiones de autorización, SAML y XACML. En particular los mensajes que se intercambian siguen el perfil del protocolo SAML para XACML. En la Figura 5.47 se resume el mecanismo de control de acceso implementado junto a los estándares y mensajes de comunicación que se aplican para cada elemento.

Tanto los gestores de acceso a los recursos (*AccessManager*) como los de contexto (*ContextHandler*) se distribuyen por todo el sistema y su número depende de las necesidades de cada POVO. Aunque un solo gestor de acceso y otro de contexto podrían cumplir con las necesidades de todo el sistema de control de acceso de una POVO, el cuello de botella en que se convertirían reduciría drásticamente la efectividad del sistema y anularía por completo las ventajas de la distribución de componentes. Por tanto, siempre que se cumpla con que todos los recursos están protegidos por algún gestor de acceso y éstos sean capaces de derivar las peticiones de acceso a gestores de contexto para la toma de decisiones, el número de elementos (así como su ubicación física) puede ser completamente variable. Como se expuso en el punto de vista de Ingeniería, el uso del estilo arquitectural SOA permite que los distintos componentes publiquen sus capacidades en registros semánticos que otros componentes utilizarán para descubrirlos y acceder a ellos y así se resuelve la multiplicidad de gestores de acceso y contexto.

Todos los gestores están implementados como servicios software en continua ejecución a la espera de interceptar peticiones de acceso. En concreto el gestor de acceso recibe una petición de un usuario y debe derivarla a un gestor de contexto para la toma de decisiones. Para ello obtiene el nombre o identificador del usuario que solicita el acceso, el del recurso protegido y la acción que el primero

pretende ejecutar. Con estos campos construye una petición estandarizada *XACMLAuthzDecisionQuery* y la envía al gestor de contexto que considere más adecuado quedando a la espera de la respuesta con la decisión. Una vez que la recibe, comprueba el resultado de la misma y la comunica al usuario haciendo cumplir tanto un permiso como una prohibición de acceso. Si además existiesen obligaciones que cumplir antes, durante o después del acceso, el gestor de acceso contactará (a través del registro semántico de servicios) con el servicio de obligaciones que mejor se ajuste a sus necesidades, esto es, aquel que pueda hacer efectivas las acciones obligatorias.

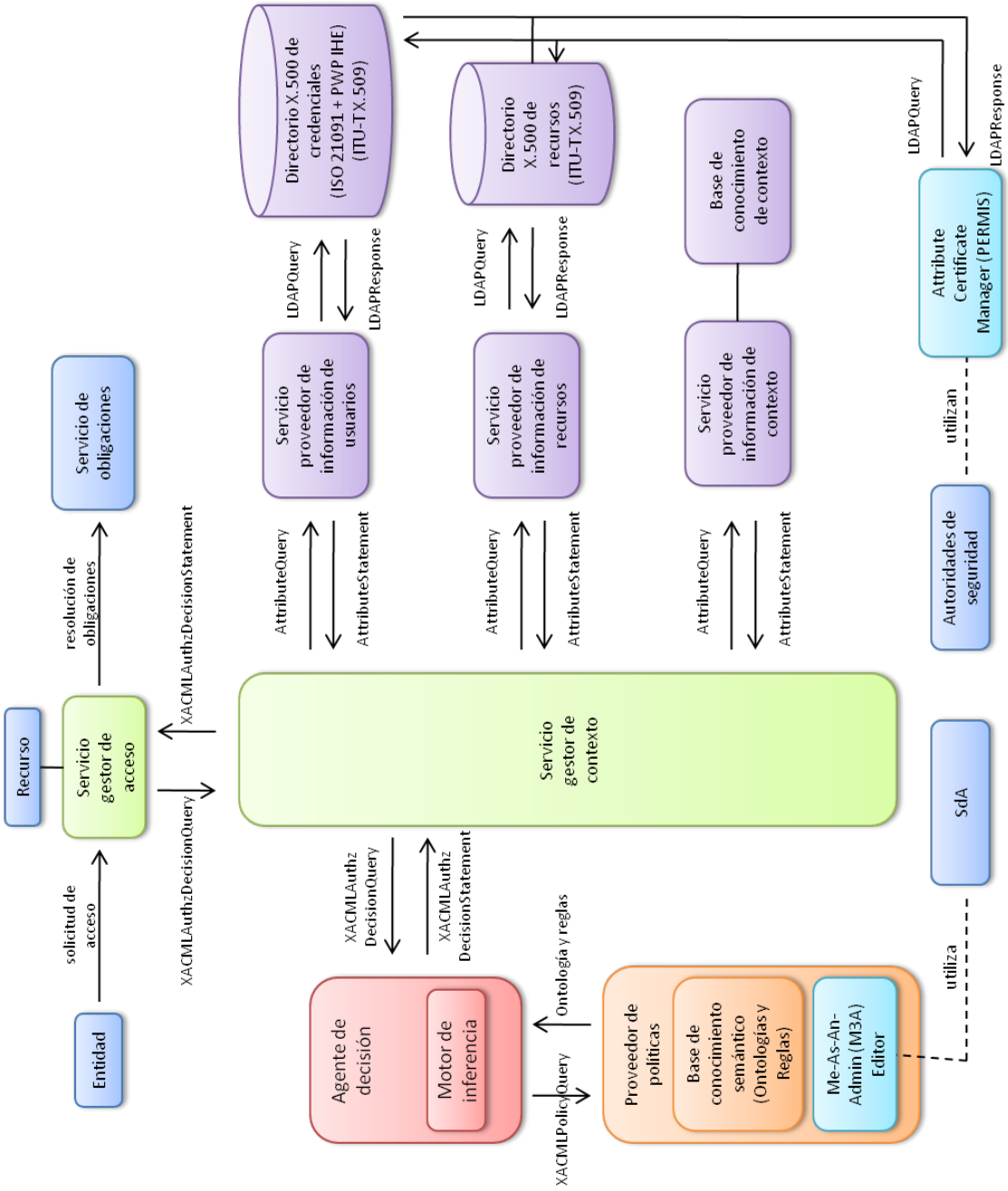

Figura 5.47. Implementación del mecanismo de control de acceso de las POVO

Por su parte, el gestor de contexto recibe la petición de decisión de autorización de los gestores de acceso (*XACMLAuthzDecisionQuery*) y la deriva directamente al agente de decisión más adecuado. No es necesario en este punto recabar más información para la petición puesto que podría darse el caso en el que se pudiera resolver sólo con el identificador del usuario, recurso y acción. El agente de decisión devolverá la respuesta en un mensaje *XACMLAuthzDecisionStatement* y el gestor de contexto comprobará si ha podido ser resuelta. Si se cuenta con una prohibición o permiso de acceso, entonces ese mensaje se pasa directamente al gestor de acceso que solicitó la decisión sin modificarla. En caso de que no se haya tomado una decisión (se obtiene como resultado *Indeterminate* o *NotApplicable*), el gestor de contexto es el encargado de recabar información sobre el usuario, recurso y contexto para completar la petición e intentar resolverla. Para ello, consultará un registro semántico de servicios y contactará con los proveedores de información más adecuados, enviando a cada uno un mensaje *AttributeQuery* solicitando información sobre, según el caso, el usuario, el recurso o el contexto. Queda entonces a la espera de recibir los mensajes *AttributeStatement* de los proveedores y, una vez recibidos, crea una nueva petición de decisión de autorización (*XACMLAuthzDecisionQuery*) con toda la información recibida y la envía al agente de decisión. Vuelve a esperar la respuesta y esta vez, sea cual sea ésta, la devuelve al gestor de acceso para que la comunique al usuario. Si aún es una respuesta indeterminada, se resolverá denegando el acceso que es la decisión más conservadora.

En segundo lugar tenemos los puntos de administración de políticas (en plural porque puede haber un número indeterminado de ellos) los cuales recogen políticas de control de acceso definidas por la autoridad de seguridad (en este caso el administrador de la POVO, SdA). Siguiendo el mecanismo semántico de control de acceso los agentes de decisión deben poder solicitar la ontología aplicable al escenario que se está evaluando (puede ser la ontología federada junto con las locales relevantes para el usuario que solicita el acceso y el recurso accedido) así como las políticas concretas como reglas SWRL. Por un lado el punto de administración de políticas sirve a estos clientes proporcionándoles la información que precisan y por otro cuenta con un servicio de usuario final para que el SdA introduzca las políticas de control de acceso que considere necesarias. La Figura 5.48 representa el conjunto de elementos que forman parte de un proveedor de políticas.

Se ha hecho especial hincapié en el desarrollo de aplicaciones que potencien la usabilidad de las soluciones. Como punto de acceso del SdA al mecanismo de control de acceso se presenta la aplicación M3A (*Me-As-An-Admin*) que guía a la persona a través de la especificación de políticas personales para el control de acceso a sus recursos distribuidos. La aplicación M3A ha sido desarrollada en el ámbito de los lenguajes específicos de dominio (DSL) y la ingeniería asistida por modelos (MDE), aspectos que describimos brevemente a continuación. Los DSL constituyen un área cada vez más popular dentro de la comunidad MDE a causa de su simplicidad comparada con los lenguajes de propósito general y su especial consideración del dominio de interés. Un DSL está basado en un metamodelo (la sintaxis abstracta que especifica los conceptos del lenguaje y sus relaciones) y una sintaxis concreta (que permite a los usuarios crear modelos conformes al metamodelo). Entre las herramientas disponibles

que soportan el desarrollo de DSL, el Marco de Modelado Gráfico de Eclipse (*Eclipse Modeling Framework,* GMF) [209][210] es una de las más prometedoras.

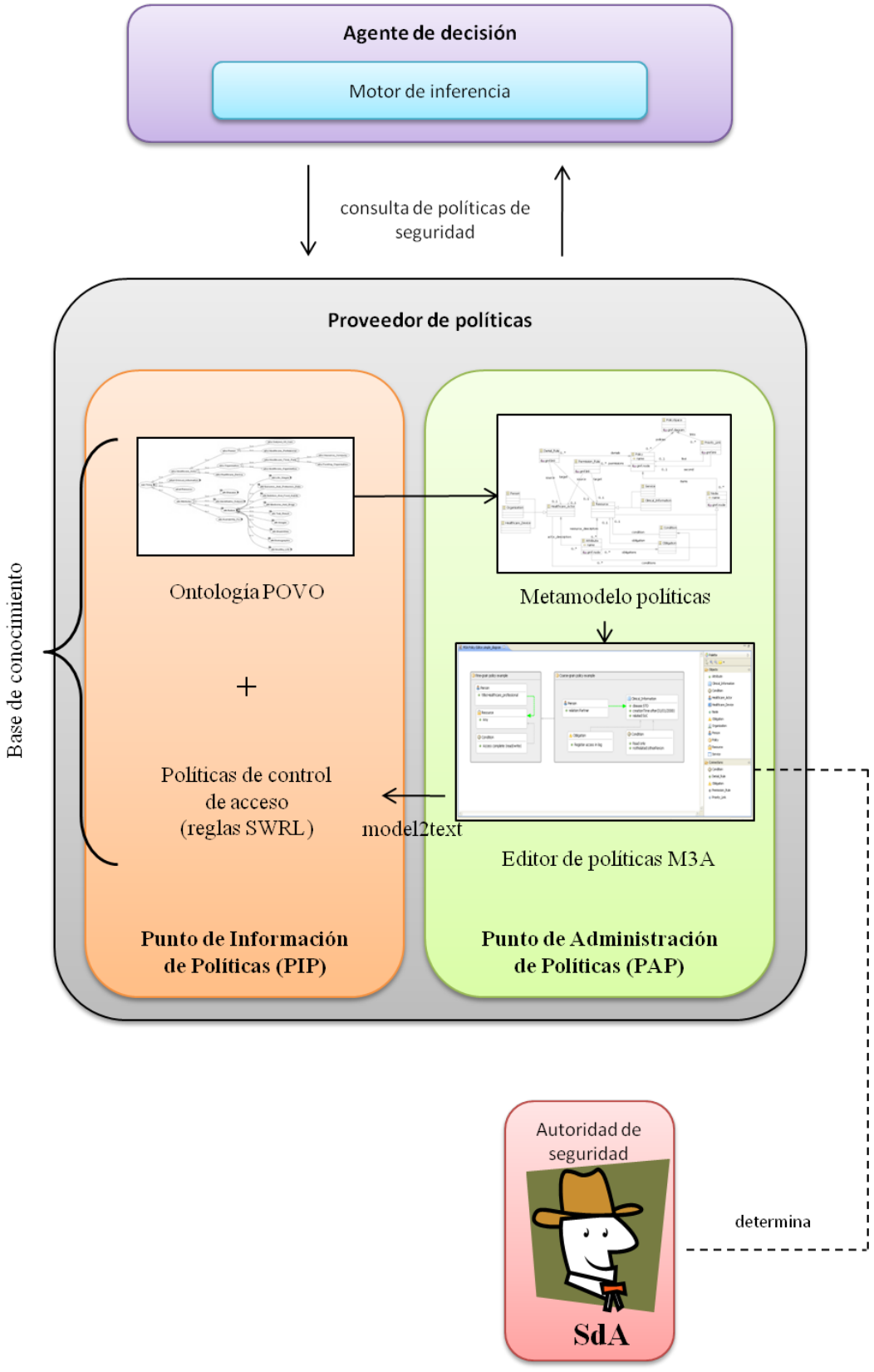

Figura 5.48. Componentes del proveedor de políticas (puntos de información y administración)

Para aliviar algunas debilidades en términos del elevado esfuerzo requerido para los desarrollos con esta herramienta, algunos plug-ins han sido creados para soportar funcionalidades avanzadas y hacer transparentes algunas tareas de bajo nivel. Estos complementos son el Marco de Modelado Gráfico (*Graphical Modeling Framework,* GMF) [211] y EuGENia [212]. La filosofía MDE propone trabajar con modelos a ciertos niveles de complejidad y transformarlos en otros de diferentes niveles. Además de la transformación entre modelos, también existen herramientas que permiten transformaciones de modelos a texto como MoFScript [213] o la iniciativa Model-to-Text (M2T) [214]. Éstas utilizan un archivo que especifica las transformaciones utilizando plantillas e indicando qué pieza de código o texto corresponde con cada elemento del modelo.

El editor permitirá que un usuario (un SdA sin conocimientos tecnológicos avanzados) especifique de forma sencilla modelos que representen sus preferencias sobre el control de acceso a sus recursos. Los modelos serán automáticamente transformados a texto (en concreto, a reglas SWRL) utilizando M2T e incluidos en la base de conocimiento. En el presente trabajo se ha desarrollado un metamodelo para el editor de políticas de control de acceso a partir de la ontología POVO. Debido al amplio rango de conceptos que la ontología encierra y por simplicidad, se ha trabajado en una versión reducida de la ontología manteniendo la sub-ontología de seguridad (esencial para la edición de políticas de control de acceso) y la de actores sanitarios. El metamodelo ha sido creado en Emfatic [215], una sintaxis textual para Ecore, y la Figura 5.49 muestra los principales conceptos y relaciones. Este metamodelo sirve como prueba de concepto para mostrar la usabilidad de una solución orientada a los SdAs y en un escenario real debería ser extendida con la ontología POVO completa. Como se muestra en la figura el concepto principal es *BasicPolicy* que contiene nodos (*nodes*) y enlaces (*links*). Los nodos son utilizados para la edición de políticas e incluyen: actor sanitario (*Healthcare_actor,* que solicita el acceso), recurso (*Resource*, al que se pretende acceder) y obligaciones (*Obligations*) y condiciones (*Conditions*) que limitan e imponen restricciones al acceso. Cada nodo puede contener atributos (*Attributes*) que en esta primera versión son descripciones textuales. La relación entre un actor y un recurso se establece a través de los enlaces *Auth+* y *Auth-* especificando si el acceso se permite o deniega a priori (ya que las condiciones tendrán que ser satisfechas para garantizar el acceso). Además una relación de prioridad (*Priority*) entre políticas permite determinar preferencias en el orden de comprobación de las mismas. Este metamodelo ha facilitado el desarrollo del editor M3A mediante las herramientas EuGENia.

El siguiente componente es el editor de políticas M3A el cual se ha diseñado para soportar SdAs con variedad de capacidades. La definición de políticas se realiza utilizando modelos que serán traducidos a políticas inteligibles por el sistema, esto es, reglas SWRL. La Figura 5.50 presenta una muestra del editor M3A el cual incluye una paleta con los elementos definidos en el metamodelo y el SdA sólo tiene que arrastrarlos al espacio de trabajo. En esta figura se muestra un ejemplo en el que se utilizan dos políticas. La política superior especifica la regla generalista que se utilizó anteriormente como ejemplo (es decir, cualquier profesional sanitario puede acceder a cualquier recurso para lectura y escritura sin restricciones ni condiciones) mientras que la política inferior corresponde al ejemplo de regla detallada

que permite la consulta de información relacionada con enfermedades de transmisión sexual para la esposa del SdA. Una condición limita que no se puede acceder a información que identifique a una persona diferente del propio SdA. Además se obliga a que, para conceder el acceso, éste sea registrado en un log de eventos. En este ejemplo existe una prioridad de la política específica sobre la generalista de manera que aunque la esposa del SdA sea un profesional sanitario no podrá consultar la información sobre enfermedades de transmisión sexual entre el SdA y su anterior pareja.

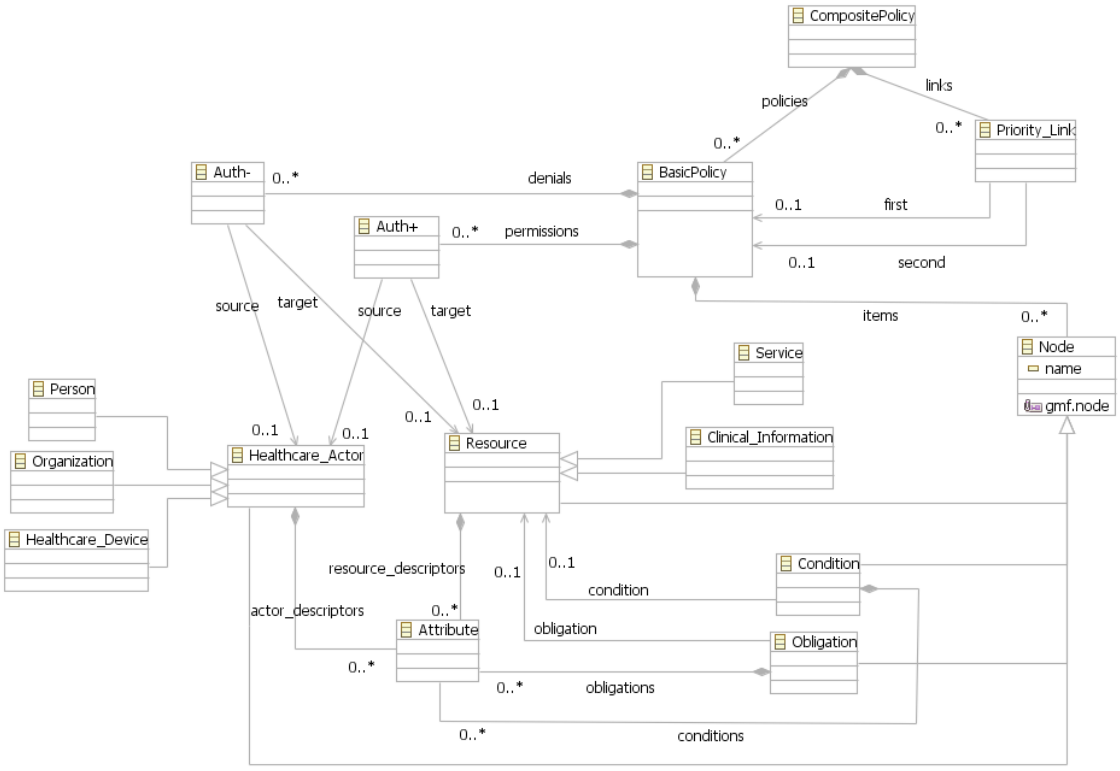

Figura 5.49. Metamodelo simplificado para el editor de políticas de control de acceso

Para la evaluación de usabilidad del editor M3A, una primera fase de validación ha sido realizada con los compañeros de trabajo del autor de modo que podrían considerarse como usuarios con aptitudes tecnológicas. La aceptación ha sido en general alta y los resultados han permitido definir algunas restricciones para reducir errores comunes que llevan al mal funcionamiento de la aplicación. Algunas de estas restricciones identificadas han sido: que no puede establecerse un enlace de prioridad entre una política y ella misma, que sólo puede haber una relación entre un actor y un recurso, que las políticas no pueden quedar sin nombre, etc. Estas y otras restricciones han sido especificadas a través del Lenguaje de Validación Epsilon (*Epsilon Validation Language*, EVL) [216], un lenguaje similar a OCL.

El último paso del proceso consiste en la transformación de los modelos a texto y para ello se han definido reglas de transformación utilizando herramientas del proyecto M2T que permiten obtener, desde las instancias del metamodelo creadas por el SdA, un archivo de texto con las correspondientes

políticas expresadas como reglas SWRL. Estas políticas se incorporan en la base de conocimiento (Figura 5.48) para la posterior consulta por parte del agente de decisión.

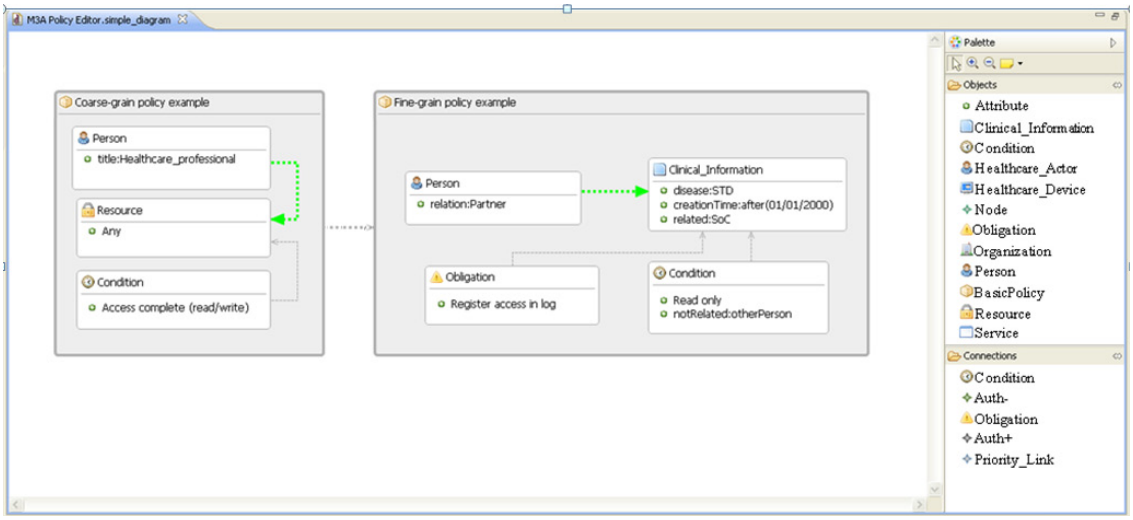

Figura 5.50. Editor de políticas Me-As-An-Admin (M3A) y dos políticas de ejemplo

El proveedor de políticas ha sido implementado como un servicio que responde a las peticiones de los agentes de decisión enviándoles el conjunto de políticas que se aplican en una POVO específica. Las peticiones serán recibidas como mensajes estandarizados *XACMLPolicyQuery* y las responderá enviando por un lado el conjunto de políticas legislativas y por otro las que hayan sido definidas por el usuario y los propietarios de los recursos. Como un agente de decisión puede dar servicio a varias POVOs, se ha optado por que le indique al servicio proveedor de políticas la POVO de interés y éste le responda con una referencia a la existencia de la ontología de aplicación. Con la ontología y las reglas el agente de decisión podrá efectuar los procesos de inferencia pertinentes para tomar una decisión sobre el acceso al recurso.

Como se ha descrito anteriormente el mecanismo de autorización se basa en múltiples fuentes de información distribuidas a través de dominios y que aportan conocimiento aplicable en la toma de decisiones de control de acceso. Principalmente se han descrito tres categorías de servicios en función del tipo de información que manejan: de usuarios, de recursos y de entorno o contexto. Cada una de ellas ha sido implementada de forma diferente atendiendo a los requisitos específicos que impone la sensibilidad de la información en cada caso. Al margen del almacenamiento de la información, la lógica funcional que siguen los proveedores de información es la misma para los tres casos:

- Los servicios proveedores de información publican sus capacidades en un registro semántico y son accedidos por gestores de contexto (*Context Handler*). Este acceso se hace a través del protocolo SAML y su perfil para XACML, es decir, reciben una petición de atributos (*AttributeQuery*) y responden con una declaración de atributos (*AttributeStatement*).

- Los servicios proveedores de información de usuarios y recursos cuentan con funcionalidad para que los administradores de seguridad correspondientes puedan crear, modificar o eliminar información. La información de contexto es obtenida a través de mecanismos internos a la plataforma sin que intervenga ninguna persona en su definición.

A continuación se detallan las implementaciones de los tres tipos de proveedores de información comentando, cuando es necesario, las herramientas que se han utilizado en cada caso.

- **Los servicios proveedores de información de contexto**

Los atributos del entorno o contexto son aquellos que no están asociados ni con el recurso, ni con la acción ni con ninguno de los sujetos de la petición de acceso. Esta información se recoge de forma automática desde la plataforma e incluye aspectos temporales como la fecha u hora, la localización física de los accesos, la autoridad de seguridad de los dominios, etc. La implementación se ha llevado a cabo creando una base de conocimiento semántica (ontologías OWL) a la que se accede cuando se recibe una petición desde un gestor de contexto. De esa consulta se sacan aquellos atributos concretos que se están solicitando y se ejecutan consultas a través del lenguaje SPARQL [86] para recuperar sus valores de la ontología. Una vez obtenidos los valores se compone una respuesta a la petición, es decir, un *AttributeStatement* y se devuelve al gestor de contexto que solicitó la consulta. La implementación de la funcionalidad semántica se ha realizado a través del lenguaje Jena [190].

- **Los servicios proveedores de información de recursos**

Los atributos que caracterizan los recursos (y cuyo papel es clave en el control de acceso desarrollado) son otorgados por autoridades de seguridad o, en su defecto, por sus propietarios. Esto hace que un mismo recurso pueda poseer diferentes conjuntos de atributos en función de la entidad que los haya dispensado lo que conlleva una incertidumbre a la hora de utilizar unos u otros. Por tanto se hace necesario un mecanismo por el cual se pueda siempre identificar a la entidad que otorgó cada atributo a cada recurso y así poder establecer niveles de confianza sobre los atributos en función de la confianza en la entidad dispensadora. El mecanismo que se ha utilizado es el de los Certificados de Atributos (*Attribute Certificate*), parte de la norma ITU-T X.509 [94]. Así un certificado contiene un conjunto de atributos para un recurso e identifica además la entidad que lo ha creado (y otorgado los atributos al recurso). Cuando un gestor de contexto reciba los atributos sabrá quién los dispensó y eso podrá influir en la confianza que se tiene en esa información. Además los certificados cuentan con mecanismos intrínsecos de seguridad para asegurar la integridad de los datos que contienen. Los certificados de atributos se almacenan en directorios conformes a la norma ITU-T X.500 [217] y se accede a ellos a través del protocolo LDAP (*Lightweight Directory Access Protocol*) [145]. La razón por la que se utilizan directorios en lugar de bases de datos tradicionales es porque la información contenida en un directorio normalmente se consulta con mayor frecuencia que se modifica. Como consecuencia los directorios no implementan normalmente los complicados esquemas para transacciones o esquemas de reducción que

las bases de datos utilizan para llevar a cabo actualizaciones complejas de grandes volúmenes de datos. Por el contrario, las actualizaciones en un directorio son usualmente cambios sencillos. Los directorios proporcionan una respuesta rápida a operaciones de búsqueda y consulta. Pueden tener capacidad de replicar información de forma amplia con el fin de aumentar la disponibilidad y fiabilidad y a la vez reducir el tiempo de respuesta. El servicio de directorio LDAP se basa en un modelo cliente-servidor y así uno o más servidores LDAP contienen los datos que conforman el árbol de directorio LDAP o base de datos troncal, el cliente LDAP se conecta con el servidor LDAP y le hace una consulta. El servidor contesta con la respuesta correspondiente o bien con una indicación de donde puede el cliente hallar más información. No importa con qué servidor LDAP se conecte el cliente ya que siempre observará la misma vista del directorio; el nombre que se le presenta a un servidor LDAP hace referencia a la misma entrada a la que haría referencia en otro servidor LDAP. En la implementación de los proveedores de información de recursos y usuarios se ha utilizado la implementación libre OpenLDAP [218].

El funcionamiento de los proveedores de información de recursos es como sigue:

- Un gestor de contexto envía una petición de atributos sobre un recurso al proveedor de información que considere más adecuado. Previamente, los proveedores de información deben haber publicado sus capacidades en un registro semántico para ser descubiertos.

- El proveedor extrae de la petición el identificador del recurso cuyos atributos se solicitan y utiliza el protocolo LDAP para acceder al directorio que contiene los certificados de recursos.

- Una vez recuperado el certificado o certificados del recurso, y como la comunicación con el gestor de contexto debe utilizar el protocolo SAML, se extraen del certificado los atributos y se compone una respuesta del tipo *AttributeStatement*, la cual se envía de vuelta al gestor de contexto.

Lo único que resta es especificar cómo se lleva a cabo la dispensación de atributos a los recursos, es decir, cómo las entidades de seguridad crean los certificados de atributos y los almacenan en el directorio LDAP. Para ello se ha utilizado la herramienta Gestor de Certificados de Atributos (*Attribute Certificate Manager,* ACM) que forma parte del sistema de autorización PERMIS [219]. PERMIS es un sistema de autorización basado en políticas de acceso que implementa una versión mejorada del modelo estándar de control de acceso basado en roles (RBAC). PERMIS permite la asignación de roles y atributos a los usuarios ofreciendo seguridad mediante el uso del algoritmo de encriptación de clave pública y de atributos certificados según la norma X.509. El ACM permite crear certificados X.509 que contienen atributos que asignan credenciales de acceso a los usuarios o categorizan los recursos. Estos certificados son emitidos por un administrador del sistema y firmados siguiendo el algoritmo de clave pública.

- **Los servicios proveedores de información de usuarios**

Para el almacenamiento de información de usuarios se ha considerado una doble alternativa. En primer lugar se mantiene el esquema de los proveedores de información de recursos visto en el apartado anterior, esto es, certificados de atributos (pero ahora para usuarios) otorgados y firmados por entidades de seguridad a través de la herramienta ACM de PERMIS. El directorio se implementa con OpenLDAP y sigue el protocolo LDAP de comunicaciones.

La razón por la que se incluye una segunda alternativa de almacenamiento es porque en nuestro escenario se pueden aplicar dos recomendaciones en caso de no utilizar certificados de atributos: la norma ISO 21091 Informática sanitaria – Servicios de directorio para seguridad, comunicaciones e identificación de profesionales y pacientes (*Health informatics - Directory services for security, communications and identification of professionals and patients*) [220] y el perfil del IHE de Páginas Blancas de Personal (*Personnel White Pages,* PWP) [143]. La primera especifica un esquema de directorio para el almacenamiento de información relativa a actores sanitarios en directorios conformes a la norma ITU-T X.500. Se especifica, por tanto, el espacio de nombrado que se debe aplicar así como un conjunto de entradas de directorio y sus correspondientes atributos obligatorios y opcionales. Algunas de estas entradas son: Consumidor de asistencia sanitaria (*Health care consumer*), Profesional de asistencia sanitaria (*Health care professional*), Empleados (*Employees*), Organización sanitaria regulada (*Regulated health care organization*) o Rol estándar sanitario (*Health care standard role*). Por otro lado, los perfiles del IHE buscan facilitar la interoperatividad de sistemas dentro del dominio sanitario a partir de la aplicación de estándares y recomendaciones. El perfil PWP cubre el acceso a directorios básicos de información sobre trabajadores en la organización sanitaria así como la información que puede estar almacenada en el directorio. El perfil no tiene en cuenta a pacientes ni otros individuos que no tomen parte en los procesos sanitarios ni define mecanismos de control de acceso o auditoría. Las transacciones entre los consumidores de páginas blancas y los servicios de directorio deben realizarse a través del protocolo LDAP. Aplicando estas dos recomendaciones se obtiene un esquema de directorio X.500 con entradas para todos los actores, organizaciones y roles y al que se debe acceder a través del protocolo LDAP. La definición formal que se realiza en estas normas sobre las entradas del directorio y sus atributos (definiendo sus identificadores universales) es una apuesta por la normalización y potenciación de la interoperatividad entre sistemas heterogéneos si bien es cierto que se pierde flexibilidad ya que todas las entidades de seguridad y dominios deben limitarse a utilizar el conjunto de atributos especificado. Por ello hemos considerado interesante implementar esta alternativa a la vez que permitimos el uso de certificados de atributos, con éstos perteneciendo a ontologías locales que pueden ser mapeadas a la ontología federada.

Al margen de los servicios proveedores de información, uno de los componentes más importantes del mecanismo de control de acceso es el agente de decisión el cual se ha implementado como un servicio que publica sus capacidades y recurre a los registros para descubrir los servicios proveedores de políticas que le interesan. Recibe desde gestores de contexto mensajes estandarizados de petición de decisión de acceso (*XACMLAuthzDecisionQuery*) y, una vez tomada la decisión, responde con los

correspondientes mensajes *XACMLAuthzDecisionStatement.* Una vez que recibe una petición de decisión, envía un mensaje *XACMLPolicyQuery* al servicio proveedor de políticas de su elección y espera la respuesta que se compondrá del conjunto de reglas legislativas, el de reglas de usuarios y una referencia a la ontología de aplicación en la POVO. El proceso interno que sigue este servicio para tomar una decisión es el que se muestra en la Figura 5.46. Primero se recoge la ontología a partir de la referencia recibida y se combina con las políticas legislativas. Se lleva a cabo un proceso de inferencia a través del razonador Pellet [84] y se realiza una consulta para comprobar si existen relaciones de permiso o prohibición entre el usuario y el recurso implicados en el acceso. De ser así, se compone la respuesta y envía al gestor de contexto. En caso contrario se procede de nuevo combinando ahora las políticas definidas por el SdA y los propietarios de los recursos con la ontología. Se realiza de nuevo la inferencia y se consultan las propiedades entre el usuario y el recurso. Ahora sí se compone una respuesta aunque no se haya podido tomar la decisión y se envía al gestor de contexto que solicitó el acceso. Sólo queda mencionar que la implementación de la funcionalidad semántica de este servicio se ha llevado a cabo a través del lenguaje Jena.

Finalmente el servicio de obligaciones tiene el objetivo de hacer cumplir las acciones que pueden condicionar un permiso de acceso. Las obligaciones son acciones que deben ser realizadas antes, después o durante el cumplimiento de la decisión de autorización. Muchas obligaciones serán específicas de aplicación pero algunas pueden ser independientes como por ejemplo la inclusión de la decisión de acceso en un registro de auditoría o la notificación al usuario de que a alguien se le ha concedido acceso a su información personal. De acuerdo con el modelo XACML cada obligación tiene un identificador único. El servicio de obligaciones sabrá qué obligaciones puede manejar ya que estará configurado con el conjunto de identificadores de las obligaciones que soporta. Una vez que recibe las obligaciones, ejecutará las que pueda y derivará aquellas que estén fuera de su capacidad a otros servicios de obligaciones. Si se da un error en la ejecución de las obligaciones se comunica al gestor de acceso para denegar el acceso del usuario al recurso.

6.4.10 La formalización del punto de vista de Tecnología

El punto de vista de Tecnología se centra en el despliegue real de tecnología para implementar el sistema ODP. Una especificación de tecnología define objetos de tecnología (productos hardware, software y de red) a partir de diagramas de configuración de nodos. Debido a que el mecanismo de control de acceso presentado en este trabajo posee una enorme flexibilidad de componentes no existe un único escenario de despliegue que pueda ser formalizado a través de elementos hardware y software. Por ello recurrimos a una visión genérica del sistema y a un ejemplo de muestra de un escenario específico. Unida a esta formalización está toda la descripción del software de implementación de cada elemento que se ha introducido anteriormente. En la Figura 5.51 se presenta el esquema genérico de configuración de los nodos que sigue el esquema de la Figura 5.11. Todos los

nodos utilizados en la formalización del punto de vista de Tecnología serán de alguno de los cuatro tipos siguientes:

- Consumidor de servicio (*Service consumer*): hace uso de las capacidades de los proveedores de servicios y puede actuar en representación de una persona.

- Proveedor de servicio (*Service provider*): publica su funcionalidad y modo de acceso a ella en un registro semántico y resuelve peticiones de los consumidores.

- Registro semántico (*Semantic Registry*): almacena los perfiles semánticos (funcionalidad y modo de acceso) de los proveedores y permite que los consumidores descubran los proveedores que mejor se ajustan a sus necesidades.

- Red de área extensa (*WAN*): representa el soporte de comunicaciones para el enlace entre los componentes distribuidos del sistema. Se consideran exclusivamente redes de área extensa aunque podrían incluirse también aquellas de área local. Todos los componentes están conectados a este tipo de objetos.

Una vez que se ha descrito el esquema abstracto del sistema de control de acceso, se representa en la Figura 5.52 un escenario concreto como ejemplo de despligue de sistema. Este sistema se despliega sobre una red de área extensa llamada WAN1 y con un cliente, un recurso y un administrador. No hay multiplicidad en los componentes del sistema (es decir, sólo existe un elemento de cada tipo) y sólo se cuenta con un registro semántico. Aquellos elementos que simultáneamente proveen de servicios y son consumidores de los expuestos por otros (estos son *ContextHandler, DecisionAgent* y *AccessManager*) han sido separados en un componente como proveedor de servicio y otro como consumidor. Como esta doble funcionalidad, en general, estará agrupada en el mismo objeto tecnológico, cada par de objetos se ha unido indicando que pueden comunicarse de forma directa sin hacer uso de la red de área extensa.

6.4.11 Correspondencias entre el punto de vista de Ingeniería y el de Tecnología

El punto de vista de Tecnología sólo tiene correspondencias con el punto de vista de Ingeniería. Según la especificación de RM-ODP un conjunto de uno o más objetos de tecnología corresponden a un objeto de ingeniería e implementan la funcionalidad especificada en el objeto de ingeniería de forma tecnológica. Como no existe un escenario tecnológico concreto de aplicación no se pueden establecer las correspondencias entre objetos.

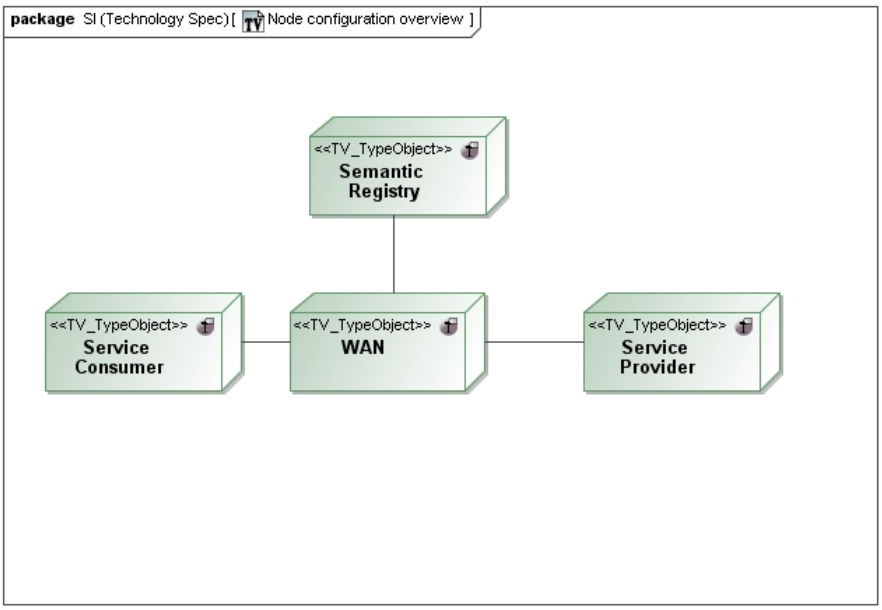

Figura 5.51. Esquema de alto nivel de la configuración de nodos

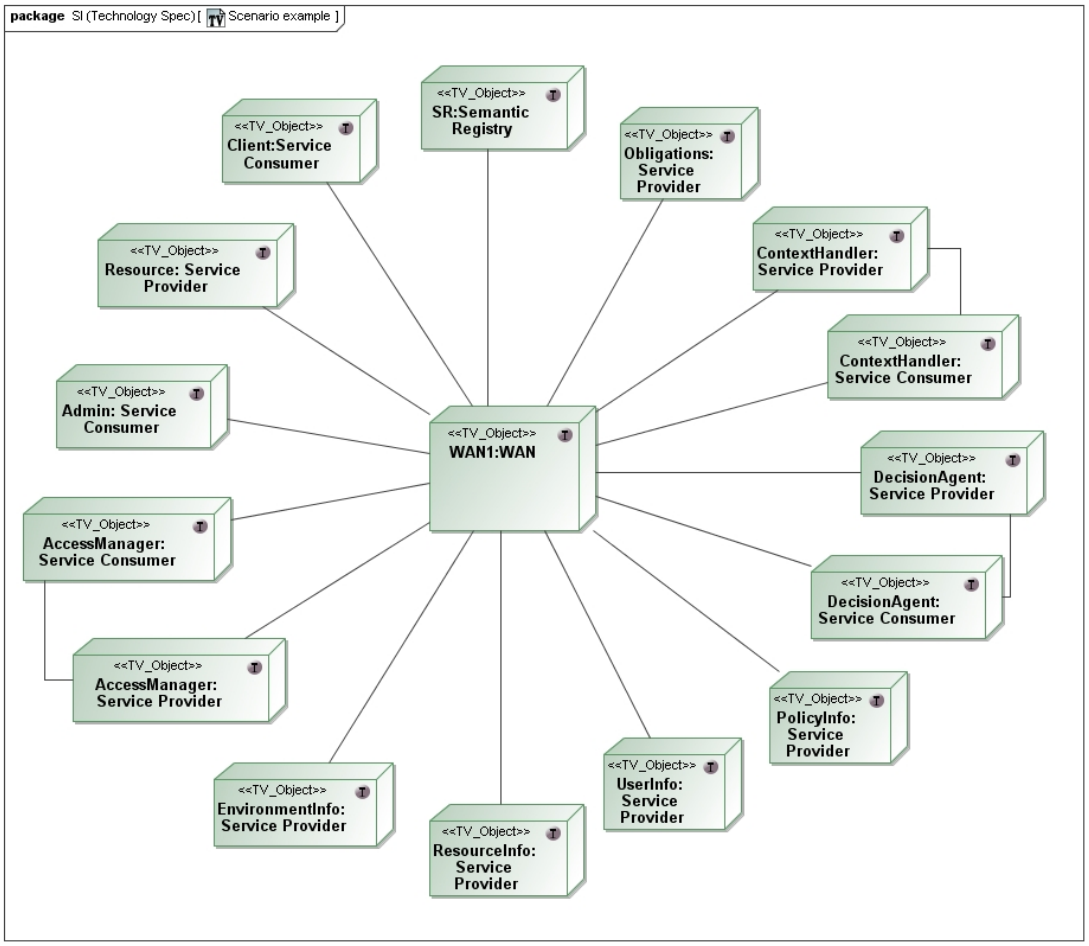

Figura 5.52. Ejemplo de configuración de nodos para un escenario específico

Sección 4. Conclusiones y Anexos

Capítulo 6. Conclusiones y Línea de investigación

1. Conclusiones

Al comienzo de este trabajo de Tesis Doctoral se expusieron las hipótesis en las que se basaba la investigación realizada. Éstas eran:

- Los escenarios centrados en el SdA poseen un enorme potencial para revolucionar la práctica asistencial haciéndola más eficiente y avanzada así como reduciendo los costes asociados. Los sistemas distribuidos son el paradigma de diseño que mejor se ajusta a estos nuevos escenarios.

- Ante la heterogeneidad de los sistemas y tecnologías existentes en la actualidad la normalización es la clave para potenciar la interoperatividad de las soluciones y resolver el problema de los sistemas fragmentados facilitando la construcción de sistemas avanzados que reutilizan las capacidades de otros.

- El rol activo de los SdAs en su salud es un aspecto clave de los nuevos escenarios de asistencia. La tecnología permite capacitar al SdA para que se implique en los procesos relacionados con su salud en diversos ámbitos. Uno de éstos es el de administración de los permisos de acceso a información sanitaria y personal, es decir, autorización y control de acceso.

Tras la realización de este trabajo, que ha supuesto una profunda investigación en varios ámbitos de conocimiento así como el diseño y desarrollo de sistemas específicos como pruebas de concepto sobre las hipótesis, se pueden extraer las siguientes conclusiones:

1. El potencial que la normalización tiene para construir sistemas y servicios interoperables y reutilizables es una realidad. Las iniciativas de normalización en cada disciplina y ámbito de conocimiento que proporcionan guías, esquemas y metodologías de desarrollo facilitan la adopción de modelos estándar. En este trabajo la adopción de estilos arquitecturales estándar, modelos de información de referencia, esquemas normalizados de seguridad, etc., ha posibilitado la construcción de sistemas abiertos con garantías de poder ser reutilizados en futuros esfuerzos o extendidos con funcionalidades avanzadas. Por otro lado, fruto de la experiencia adquirida con este trabajo se puede afirmar que la adopción de estándares requiere un sobreesfuerzo en el desarrollo de sistemas ya de por sí complejos. A esto hay que añadir la complejidad que supone la existencia de iniciativas normalizadoras solapadas (o incluso en conflicto) que requieren esfuerzos de armonización por parte de los propios interesados. En resumen, la adopción de estándares para el desarrollo de sistemas es un requisito de diseño costoso en tiempo y esfuerzo si bien otorga a los sistemas finales un valor añadido que posibilita su integración en entornos heterogéneos, la reutilización de sus capacidades para proporcionar servicios avanzados y una larga vida útil.

2. El trabajo realizado de análisis, reestructuración y extensión de la norma ISO/EN 12967 (HISA) nos lleva a concluir que dicho estándar tiene un enorme potencial para convertirse en una referencia internacional para el desarrollo de arquitecturas en el dominio sanitario. Sin embargo, para

alcanzar este estatus es necesario que HISA esté correctamente integrada con otras normas del mismo ámbito, siguiendo el mismo razonamiento sobre normalización del punto anterior. El potencial de la norma reside principalmente en su flexibilidad para dar cabida (en forma de extensiones a la norma) a otros esfuerzos normativos en cualquier ámbito relacionado con la asistencia sanitaria como se ha puesto de manifiesto en este trabajo con las extensiones de seguridad y gestión de semántica. Además se concluye que la reestructuración de la norma HISA con respecto a RM-ODP no sólo es viable sino también necesaria para la especificación formal de arquitecturas siguiendo los conceptos y lenguajes propuestos por el marco de referencia RM-ODP.

3. Los escenarios distribuidos con foco en el SdA son el paso evolutivo natural de la asistencia sanitaria dadas las actuales (y futuras) coyunturas económicas y sociales. Como se ha analizado en este trabajo la tecnología está alcanzando una gran madurez en lo que a sistemas distribuidos se refiere pero aún queda camino por recorrer para poder construir soluciones fiables y eficientes. Con motivo del paradigma POVO presentado en esta Tesis Doctoral se han identificado los principales requisitos de los escenarios distribuidos centrados en el SdA concluyendo que son numerosos y de muy diversa naturaleza. Una vez más la normalización es un pilar fundamental para satisfacer muchos de estos requisitos pero también son necesarios esfuerzos legislativos que fortalezcan el control sobre la información personal y de salud y determinen claramente la potestad que sobre su información tiene cada ciudadano.

4. Por último, el mecanismo de control de acceso diseñado y desarrollado sirve de prueba de concepto de cómo la tecnología actual puede otorgar a los individuos un papel activo en el mantenimiento de su salud y procesos relacionados sin necesidad de que tengan conocimientos tecnológicos avanzados. Este trabajo se ha basado en capacidades de gestión de semántica para la automatización de los procesos de decisión de acceso, la creación de un marco ontológico de atributos y el desarrollo de la herramienta de usuario final guiada por modelos. Aunque quedan varias funcionalidades por desarrollar completamente, este sistema permite concluir que las capacidades de gestión de semántica pueden posibilitar el desarrollo de mecanismos orientados al ciudadano y que le otorguen, por ejemplo en este caso, la potestad sobre las decisiones de control de acceso a su información y recursos sanitarios de forma real.

2. Línea de investigación

La investigación realizada durante el desarrollo de esta Tesis Doctoral ha cubierto varias disciplinas separadas como son la formalización de arquitecturas de sistemas distribuidos, los mecanismos de seguridad y control de acceso, las tecnologías semánticas y la promoción del ciudadano proactivo en los procesos relacionados con su salud. La línea de investigación iniciada por el doctorando podría resumirse como el desarrollo de métodos y modelos que faciliten la integración de sistemas distribuidos y el despliegue de escenarios centrados en el SdA como el paradigma POVO. Más concretamente se

puede hablar de aspectos de interés a profundizar a medio y largo plazo dentro de este horizonte de investigación.

En primer lugar, en esta Tesis Doctoral se ha estudiado en profundidad e identificado puntos fuertes y débiles del estándar HISA con el objetivo de alinearla con el marco de referencia ODP. El principal beneficio de este trabajo es que los modelos desarrollados permiten que el estándar ISO12967 pueda ser llevado a la práctica por diseñadores y desarrolladores ya que se dispone de una representación formal y reutilizable del esqueleto de la arquitectura normalizada. Pese a esto, y en la medida que la norma HISA sólo normaliza puntos básicos de cualquier arquitectura sanitaria, aún quedan muchos aspectos que desarrollar en profundidad. El objetivo es el de seguir apoyando esta norma y realizar contribuciones que permitan adoptarla y usarla más fácilmente. HISA debería englobar en su arquitectura todos los sistemas relacionados con el entorno sanitario. Así un interesante campo de investigación sería analizar cómo diferentes sistemas sanitarios (por ejemplo, los sistemas de Historia Clínica Electrónica con su estándar ISO 13606 o los sistemas de teleasistencia como un elemento más) se integran dentro de la arquitectura de HISA o cómo se relaciona esta norma con los perfiles de interoperatividad del IHE que están ganando popularidad internacionalmente. HISA describe aspectos normativos de las arquitecturas sanitarias y como tal debe tener en cuenta todos aquellos sistemas y componentes que se deben integrar en ellas y no considerarlos esfuerzos separados. No se trata de que HISA adopte todos los esfuerzos normativos que existan sino que se especifiquen aquellos puntos de integración entre esta norma y el resto del espectro de normas relacionadas con sistemas de interés en el dominio sanitario. Por tanto, se considera éste un interesante punto de investigación a medio plazo que permite seguir potenciando la norma HISA como referente internacional.

Otro punto de interés en la línea de investigación planteada en esta Tesis Doctoral es el de los escenarios sanitarios centrados en el SdA (como el paradigma POVO) y la promoción de éste mediante mecanismos y técnicas que le otorguen un rol proactivo en los procesos relacionados con su salud. En el presente trabajo se han descrito esfuerzos orientados a la gestión de privilegios y el control de acceso a los recursos por parte del SdA dentro de su POVO pero quedan numerosas características pendientes de investigar y desarrollar. A alto nivel la promoción y capacitación del ciudadano abarca disciplinas de seguridad como la autenticación y la auditoría (íntimamente relacionadas con el control de acceso) pero también otros muchos aspectos relacionados por ejemplo con los sistemas de apoyo a la toma de decisiones, los de aprendizaje, los que facilitan la gestión de enfermedades (auto-cuidado) o incluso los de monitorización ubicua. Todos ellos además deben ser desarrollados como soluciones orientadas y personalizadas a cualquier SdA sean cuales sean sus requisitos, necesidades y aptitudes. Además los sistemas distribuidos centrados en el SdA en general (y el paradigma POVO en particular) presentan escenarios conflictivos y aspectos legales y éticos que deberán ser analizados en detalle. Ejemplos de algunos de estos se han presentado en esta Tesis Doctoral como el conflicto entre la voluntad de un SdA y los derechos de otro (como el solapamiento de la POVO de un padre y la de su hijo), la existencia de estados de emergencia en los que se deben poder obviar las preferencias del ciudadano, la autoría de la

información como garantía de acceso permanente a dicha información, etc. Esta línea de investigación va a la par de los desarrollos tecnológicos y los complementa desde un punto de vista de aplicación en escenarios reales. Muchos de estos conflictos requerirán de la intervención legislativa y aunque desde un punto de vista de investigación no se puedan dar soluciones determinantes ante cada conflicto sí se podrán identificar aquellos escenarios que merecen un esfuerzo normativo y los sistemas y mecanismos necesarios para soportar lo que la legislación resuelva.

Otro aspecto del paradigma POVO que requiere mayor estudio y desarrollo es su nivel de infraestructura. El modelo de sistema distribuido que aúna recursos de dominios administrativos y geográficos separados plantea numerosos requisitos que en esta Tesis Doctoral hemos resuelto asumiendo transparencia de distribución. El uso de diferente software de intermediación y tecnologías de sistemas distribuidos de soporte a los escenarios centrados en el SdA es un campo extenso de investigación que será considerado en el futuro. En este trabajo se ha recurrido al paradigma de diseño SOA y la tecnología de computación en Grid como soporte al escenario POVO pero se deben considerar otras tecnologías emergentes como la computación en la nube (Cloud Computing) o las arquitecturas orientadas a eventos.

Por último, en esta Tesis Doctoral se han tratado dos dominios de conocimiento adicionales: los mecanismos de control de acceso y las tecnologías semánticas. En cuanto a la disciplina de autorización y al margen de su aplicación al escenario POVO, se ha realizado una extensa investigación sobre las iniciativas más relevantes y se ha desarrollado un marco normalizado de propósito general en base a ellas. Los mecanismos desarrollados para el paradigma POVO pueden ser aplicados a otros ámbitos e incluso podrían desarrollarse en mayor profundidad buscando hacerlos más sofisticados y complejos. Por otro lado, en este trabajo se ha puesto de manifiesto el potencial que tienen las tecnologías semánticas en numerosas aplicaciones. La gestión de semántica es un aspecto transversal a toda la arquitectura y como tal puede contribuir al desarrollo de servicios avanzados no sólo en el control de acceso sino en cualquier otro tipo de proceso. En este punto las aplicaciones de semántica que despiertan mayor interés en el doctorando son: la que facilita la personalización de los servicios y procesos de acuerdo a las necesidades, preferencias y perfiles de cada SdA, y la que potencia la interoperatividad de los sistemas a través de dominios ontológicos dispares.

Resumiendo, y aunque pueda parecer que se plantea una línea de investigación de futuro muy difusa por involucrar ámbitos muy diferenciados, el doctorando considera que todo el trabajo desarrollado y la experiencia adquirida con la presente Tesis Doctoral puede dar paso a numerosos esfuerzos investigadores que, aunque en diferentes disciplinas, contribuyen al núcleo de la futura línea de investigación: el desarrollo de métodos y modelos que faciliten la integración de sistemas distribuidos y el despliegue de escenarios centrados en el Sujeto de la Asistencia (SdA).

Capítulo 7. Bibliografía

1. Bibliografía

Para todas las referencias de recursos web la fecha del último acceso data de Septiembre del 2012.

1. WHO. Framework and Standards for Country Health Information Systems, 2ª ed. 2008.
2. ISO 12967-1, 2, 3: Health informatics – Service architecture. 2008.
3. Manso M, Wachowicz M, Bernabé M. Towards an integrated model of interoperability for spatial data infrastructures. GIS, 2009; 13(1): 43-67.
4. ISO - International Organization for Standardization. Disponible en: http://www.iso.org/iso/home.html.
5. ITU-T Rec. X.901 Information technology – Open distributed processing – Reference model: Overview. 1997.
6. Erl T. SOA: principles of service design. 5ª ed. Upper Saddle River New Jersey etc. Prentice Hall, 2009.
7. Wikipedia. Component-based software engineering, 2010.
8. McCabe FG, Estefan JA, Laskey K, Thornton D. OASIS Service Oriented Architecture Reference Architecture 1.0 Public Review Draft 1, 2008.
9. Mas A. Agentes software y sistemas multiagente: conceptos, arquitecturas y aplicaciones. Prentice Hall, 2005.
10. W3C. Web services description language (WSDL), Version 1.1, 2001.
11. Bray T, Paoli J, Sperberg-McQueen CM, Maler E, Yergeau F. Extensible markup language (XML) 1.0. W3C recommendation, 2000.
12. OWL-S: Semantic Markup for Web Services. Disponible en: http://www.w3.org/Submission/OWL-S/.
13. OWL Web Ontology Language Semantics and Abstract Syntax. Disponible en: http://www.w3.org/TR/owl-semantics/.
14. Taylor H. Event-driven architecture. Addison-Wesley, 2009.
15. Department of Defense US. DoD Architecture Framework Version 1.5, Volumen 1, 2, 3, 2007.
16. Zachman JA. A framework for information systems architecture. IBM Systems Journal 1999; 38(2.3): 454-470.
17. Hilliard R. IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems, 2000.
18. Perry DE, Wolf AL. Foundations for the study of software architecture. En Actas: SIGSOFT Softw. Eng. Notes 1992; 17(4): 40-52.
19. Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide, 2ª ed. Addison-Wesley Professional, 2005.
20. Lamb DA. IDL: sharing intermediate representations. ACM Transactions on Programming Languages and Systems, 1987; 9(3): 297-318.
21. OMG. The Common Object Request Broker: Architecture and Specification, Revision 2.0, 1995.
22. ITU: Committed to connecting the world. Disponible en: http://www.itu.int/en/pages/default.aspx.
23. ISO-15414 Information technology -- Open distributed processing -- Reference model -- Enterprise language, 2006.
24. ISO 13235-1, 2 Information technology -- Open Distributed Processing -- Trading function, 1998.
25. ISO 14769 Information technology -- Open Distributed Processing -- Type Repository Function, 2001.
26. ISO 14771 Information technology -- Open Distributed Processing -- Naming framework, 1999.
27. ISO 19793: Information technology – Open Distributed Processing - Use of UML for ODP system specifications, 2008.
28. Kutvonen L, Metso J. Services, contracts, policies and eCommunities - Relationship to ODP framework. En Actas: WODPEC 2005.

29. Kutvonen L. Using the ODP reference model for enterprise architecture. En Actas: IEEE International Enterprise Distributed Object Computing Workshop, EDOC, 2007.
30. Linington PF. Black cats and coloured birds - What do viewpoint correspondences do? En Actas: IEEE International Enterprise Distributed Object Computing Workshop, EDOC, 2007.
31. Almeida JPA, Guizzardi G. On the foundation for roles in RM-ODP: Contributions from conceptual modelling. En Actas: IEEE International Enterprise Distributed Object Computing Workshop, EDOC, 2007.
32. Object Management Group. Disponible en: http://www.omg.org/.
33. Model-Driven Architecture (MDA). Disponible en: http://www.omg.org/mda/.
34. OMG. Object Constraint Language Specification. OMG Unified Modeling Language Specification, Version 1.3, 1999.
35. ISO 19502 Information technology -- Meta Object Facility (MOF), 2005.
36. OMG. Meta Object Facility (MOF) 2.0, Query/View/Transformation (QVT) Version 1.1, 2009.
37. The Open Group. TOGAF "Enterprise Edition" Version 9, 2009.
38. The Open Group - Making Standards Work. Disponible en: http://www.opengroup.org/.
39. Bell M. Service-Oriented Modeling: Service Analysis, Design, and Architecture, 2008.
40. Scheer A, Schneider K. ARIS - Architecture of Integrated Information Systems. Handbook on Architectures of Information Systems, 2006: 605-623.
41. Bonnet P, Detavernier J, Vauquier D, Boyer J, Steinholtz E. The Praxeme Enterprise Method. Sustainable IT Architecture: The Progressive Way of Overhauling Information Systems with SOA, 2009.
42. Executive Office US. FEA Consolidated Reference Model Document Version 2.3, 2007.
43. Department of Defense US. Ministry of Defense Architectural Framework Version 1.0, 2005.
44. Kruchten PB. The rational unified process: an introduction. 2ª ed. Addison-Wesley, 2000.
45. Kruchten PB. The 4+1 View Model of architecture. IEEE Software, 1995; 12(6): 42-50.
46. Robal T, Viies V, Kruus M. The Rational Unified Process with the 4+1 View Model of Software Architecture - a Way for Modeling Web Applications. En Actas: Baltic Conference, BalticDB&IS, 2002.
47. Blobel B. Application of the component paradigm for analysis and design of advanced health system architectures. Int J Med Inf, 2000; 60(3): 281-301.
48. Hilliard R, Malavolta I, Muccini H, Pelliccione P. Realizing architecture frameworks through megamodelling techniques. En Actas: IEEE/ACM International Conference on Automated Software Engineering, 2010.
49. The Open Group. Archimate 1.0 Specification, 2009.
50. OMG. Systems Modeling Language (OMG SysML™) Version 1.1. 2008.
51. OMG. XML Metadata Interchange (XMI), 2007.
52. Network Working Group, Thurlow R. RFC 5531: RPC - Remote Procedure Call Protocol Specification Version 2. Disponible en: http://www.rfc-editor.org/rfc/rfc5531.txt, 2009.
53. Java Group. Java Remote Method Invocation (Java RMI). Disponible en: http://download.oracle.com/javase/1.5.0/docs/guide/rmi/index.html.
54. Network Working Group, Sun Microsystems I. RFC 1014: XDR - External Data Representation Standard. Disponible en: http://www.rfc-editor.org/rfc/rfc1014.txt, 2006.
55. Microsoft. Distributed Component Object Model (DCOM). Disponible en: http://msdn.microsoft.com/library/cc201989.aspx.
56. The OMG's CORBA Website. Disponible en: http://www.corba.org/.
57. Object Management Architecture. Disponible en: http://www.omg.org/oma/.

58. World Wide Web Consortium (W3C). Disponible en: http://www.w3.org/.
59. Booth D, Haas H, McCabe F, Newcomer E, Champion M, Ferris C, et al. Web Services Architecture. W3C recommendation, 2004.
60. SOAP Version 1.2 - Part 0: Primer, 2ª ed. Disponible en: http://www.w3.org/TR/2007/REC-soap12-part0-20070427/.
61. Fielding R, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P, et al. RFC 2616: Hypertext Transfer Protocol - HTTP/1.1. Disponible en: http://www.ietf.org/rfc/rfc2616.txt, 1999.
62. XML Schema Part 0: Primer, Second Edition. Disponible en: http://www.w3.org/TR/xmlschema-0/.
63. OASIS - Committees - OASIS UDDI Specifications TC. Disponible en: http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm.
64. Chumbley R, Durand J, Hainline M, Pilz G, Rutt T. WS-I Basic Profile Version 1.2, 2009.
65. The Globus Alliance. Disponible en: http://www.globus.org/.
66. IBM. Disponible en: http://www.ibm.com/us/en/.
67. Foster I, Kesselman C, Nick J, Tuecke S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, 2002.
68. OASIS Web Services Resource Framework (WSRF) TCd. Disponible en: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.
69. Foster I. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. 1st International Symposium on Cluster Computing and the Grid, 2001.
70. gLite. Disponible en: http://glite.cern.ch/.
71. UNICORE - Distributed computing and data resources. Disponible en: http://www.unicore.eu/.
72. Open Grid Forum. Disponible en: http://www.gridforum.org/.
73. WS-Management | DMTF. Disponible en: http://www.dmtf.org/standards/wsman.
74. Web Services Addressing (WS-Addressing). Disponible en: http://www.w3.org/Submission/ws-addressing/.
75. The DataGrid Project. Disponible en: http://eu-datagrid.web.cern.ch/eu-datagrid/.
76. GriPhyN Group. The Grid Physics Network (GriPhyN). Disponible en: http://www.usatlas.bnl.gov/computing/grid/griphyn/.
77. PPDD Group. The Particle Physics Data Grid. Disponible en: http://ppdg.net/.
78. OMG. Data Distribution Service for Real-time Systems, Version 1.2, 2007.
79. Neches R, Fikes R, Finin T, Gruber T, Patil R, Senator T, et al. Enabling technology for knowledge sharing. AI Magazine, 1991; 12(3): 36-56.
80. Gruber TR. A translation approach to portable ontology specifications. Knowledge Acquisition, 1993; 5(2): 199-220.
81. RDF - Semantic Web Standards. Disponible en: http://www.w3.org/RDF/.
82. DAML.org. Disponible en: http://www.daml.org/.
83. Jess, the Rule Engine for the Java Platform. Disponible en: http://www.jessrules.com/jess/index.shtml.
84. Pellet: The Open Source OWL 2 Reasoner. Disponible en: http://clarkparsia.com/pellet/.
85. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Disponible en: http://www.w3.org/Submission/SWRL/.
86. Prud'hommeaux E, Seaborne A. SPARQL Query Language for RDF, 2008.
87. Knublauch H, Fergerson RW, Noy NF, Musen MA. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. Lecture Notes in Computer Science, 2004; 3298:229-243.

88. Chappell D. Enterprise Service Bus, 2004.
89. Sun Microsystems I. Java Message Service (JMS). Disponible en: http://www.oracle.com/technetwork/java/jms/index.html.
90. Foundation for Intelligent Physical Agents. Disponible en: http://www.fipa.org/.
91. IEEE webpage. Disponible en: http://www.ieee.org/index.html.
92. FIPA Agent Communication Language Specifications. Disponible en: http://www.fipa.org/repository/aclspecs.html.
93. Yeong W, Howes T, Kille S. RFC 5246: The Transport Layer Security (TLS) Protocol, version 1.2. Disponible en: http://tools.ietf.org/html/rfc5246, 2008.
94. ITU-T Rec. X.509 The directory: Public-key and attribute certificate frameworks, 2008.
95. Klein G. Health Informatics - Service Architecture - The intended role of the EN ISO 12967 standard - An informal guide, 2009.
96. Microsoft. Connected Health Framework Architecture and Design Blueprint 2.0, 2009.
97. Lopez DM, Blobel BGME. A development framework for semantically interoperable health information systems. Int J Med Inf, 2009; 78(2):83-103.
98. HL7. HL7 Healthcare Development Framework Version 1.5, 2009.
99. HL7. HL7 Reference Information Model, 2007.
100. OpenEHR: future proof and flexible EHR specifications. Disponible en: http://www.openehr.org/ home.html.
101. Healthcare DTF. Disponible en: http://healthcare.omg.org/.
102. Ferrara FM. Specification of the DHE middleware, Version 1.0, 1995.
103. GESI. Disponible en: http://www.gesi.it/gesi/default.asp.
104. Kawamoto K, Honey A, Rubin K. The HL7-OMG Healthcare Services Specification Project: Motivation, Methodology, and Deliverables for Enabling a Semantically Interoperable Service-oriented Architecture for Healthcare. Journal of the American Medical Informatics Association, 2009; 16(6): 874-881.
105. HSSP home. Disponible en: http://hssp.wikispaces.com/.
106. IHE - Integrating the Healthcare Enterprise. Disponible en: http://www.ihe.net/.
107. Eclipse. The Eclipse Foundation open source community website. Disponible en: http://www.eclipse.org/.
108. Spackman KA, Campbell KE, Côté RA. SNOMED RT: a reference terminology for health care. En Actas: American Medical Informatics Association / AMIA Annual Fall Symposium 1997: 640-644.
109. McDonald CJ, Huff SM, Suico JG, Hill G, Leavelle D, Aller R, et al. LOINC, a universal standard for identifying laboratory observations: A 5-year update. Clin Chem, 2003; 49(4): 624-633.
110. Ferranti JM, Musser RC, Kawamoto K, Hammond WE. The Clinical Document Architecture and the Continuity of Care Record. Journal of the American Medical Informatics Association, 2006; 13(3): 245-252.
111. Rising L, Janoff NS. Scrum software development process for small teams. IEEE Software, 2000; 17(4): 26-32.
112. Cockburn A. Crystal Clear, a Human-Powered Methodology for Small Teams. Addison-Wesley Professional, 2004.
113. Beck K. Extreme Programming Explained: Embrace Change. Addison-Wesley, 1999.
114. Ambysoft I. The Agile Unified Process (AUP). Disponible en: http://www.ambysoft.com/unifiedprocess/agileUP.html.
115. Douglass BP. Real time UML workshop for embedded systems. Elsevier, 2007.
116. HL7 Service Oriented Architecture Special Interest Group (SOA SIG). SOA4HL7 - Service Oriented Architecture and HL7 v3, 2006.

117. ITU-T Rec. X.800 | ISO/IEC 7498-2:1989 Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture, 1989.
118. ISO/IEC 7498-1:1994 Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model, 1994.
119. ISO/IEC 10181-1:1996 Information technology – Open systems interconnection – Security frameworks for open system: Overview, 1996.
120. ISO/IEC 2382-8:1998 Information technology -- Vocabulary -- Part 8: Security, 1998.
121. Internet Engineering Task Force. Disponible en: http://www.ietf.org/.
122. Health Level Seven International – Homepage. Disponible en: http://www.hl7.org/.
123. W3C. RFC 4949. Internet Security Glossary, Version 2, 2007.
124. Health Level Seven Security Services Framework. Disponible en: http://www.hl7.org/library/committees/secure/HL7_Sec.html.
125. Object Management Group. OMG Security Services Specification, Version 1.8, 2002.
126. IHE. Security and Privacy through IHE. 2008.
127. Ferraiolo D, Chandramouli R, Kuhn DR. Role-based access control. 2ª ed. Artech House, 2007.
128. ISO/IEC 10181-3:1996 Information technology – Open systems interconnection – Security frameworks for open system: Access control, 1996.
129. Ferraiolo DF, Kuhn DR. Role-Based Access Controls. 15th National Computer Security Conference, 1992: 554.
130. NIST CSL Bulletin on RBAC. An introduction to Role Based Access Control, 1995.
131. ANSI INCITS 359-2004. American National Standard for Information Technology – Role Based Access Control, 2004.
132. Li N, Byun J, Bertino E. A critique of the ANSI standard on role-based access control. IEEE Security and Privacy, 2007; 5(6): 41-49.
133. Karp AH, Haury H, Davis MH. From ABAC to ZBAC: the evolution of access control models, 2009.
134. ITU-T. X.1142 Lenguaje de marcaje de control de acceso extensible (XACML 2.0), 2006.
135. XACML References and Products, Version 1.84. Disponible en: http://www.oasis-open.org/committees/download.php/27298/xacmlRefs-V1-84-1.htm.
136. Turkmen F, Crispo B. Performance evaluation of XACML PDP implementations. En Actas: ACM Conference on Computer and Communications Security, 2008.
137. Sujansky WV, Faus SA, Stone E, Brennan PF. A method to implement fine-grained access control for personal health records through standard relational database queries. J Biomed Inform, 2010; 43(5 Suppl): S46-50.
138. ITU-T. X.1141 Lenguaje de marcaje de aserción de seguridad (SAML 2.0), 2006.
139. OASIS Group. Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of XACML v2.0 for Healthcare, Version 1.0, 2009.
140. OASIS: Advancing open standards for the global information society. Disponible en: http://www.oasis-open.org/home/index.php.
141. OASIS Group. Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of Security Assertion Markup Language (SAML) for Healthcare, Version 1.0, 2009.
142. IHE. IT-Infrastructure White Paper - Access Control, 2009.
143. IHE. IT Infrastructure - Technical Framework Volume 1 (ITI TF-1) - Integration Profiles, 2010.
144. Kerberos: The Network Authentication Protocol. Disponible en: http://web.mit.edu/kerberos/.

145. Yeong W, Howes T, Kille S. RFC 1777: Lightweight directory access protocol. Disponible en: http://www.ietf.org /RFC/rfc-1777.html.
146. Healthcare Information Technology Standards – HITSP. Disponible en: http://www.hitsp.org/.
147. HITSP. SC108 - Access Control Service Collaboration, Version 1.1, 2010.
148. HITSP. T17 Secured Communication Channel Transaction, Version 1.4, 2009.
149. HITSP. C19 Entity Identity Assertion Component, Version 1.3, 2009.
150. HITSP. TP20 Access Control Transaction Package, Version 1.5, 2010.
151. HITSP. TP30 Manage Consent Directives Transaction Package, Version 1.3, 2009.
152. ASTM. E1986 -09 Standard Guide for Information Access Privileges to Health Information. Disponible en: http://www.astm.org/Standards/E1986.htm.
153. W3C. WS-Trust 1.3. Disponible en: http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.html.
154. HL7. HL7 Version 3 Standard: Role-based Access Control Healthcare Permission Catalog, Release 2, 2010.
155. ISO/TS 21298:2008 - Health informatics -- Functional and structural roles, 2008.
156. OMG. Resource Access Decision Facility Specification, Version 1.0, 2001.
157. ISO/TS 22600-1:2006 Health informatics -- Privilege management and access control-- Part 1: Overview and policy management, 2006.
158. ISO/TS 22600-2:2006 Health informatics -- Privilege management and access control-- Part 2: Formal models, 2006.
159. ISO/TS 22600-3:2006 Health informatics -- Privilege management and access control-- Part 3: Implementations, 2009.
160. ASTM E2595 -07 Standard Guide for Privilege Management Infrastructure. Disponible en: http://www.astm.org/Standards/E2595.htm.
161. ISO/DTS 14441 Health informatics -- Security and privacy requirements of EHR Systems for use in conformity assessment.
162. HSSP Security (PASS) – home. Disponible en: http://hssp-security.wikispaces.com/.
163. HL7. Privacy, Access and Security Services (PASS) - Access Control Services Conceptual Model Release 1.0, 2010.
164. Concept-level access control for the Semantic Web. En Actas: ACM Workshop on XML Security, 2003.
165. Yague MI, Mana A, Lopez J, Troya JM. Applying the semantic Web layers to access control. En Actas: 14th International Workshop on Database and Expert Systems Applications, 2003.
166. A. Naumenko. Semantic-based access control in business networks, 2007.
167. Javanmardi S, Amini M, Jalili R, Ganjisuffari Y. SBAC: Semantic Based Access Control. En Actas: 11th Nordic Workshop on Secure IT-systems, 2006: 157-168.
168. Kagal L, Finin T, Joshi A. A policy based approach to security for the Semantic Web. Lecture Notes in Computer Science, 2003; 2870: 402-418.
169. Uszok A, Bradshaw JM, Johnson M, Jeffers R, Tate A, Dalton JD, et al. KAoS policy management for semantic Web Services. IEEE Intelligent Systems, 2004; 19(4): 32-41.
170. Priebe T, Dobmeier W, Kamprath N. Supporting attribute-based access control with ontologies. En Actas: First International Conference on Availability, Reliability and Security (ARES), 2006.
171. Shen H. A semantic-aware attribute-based access control model for web services. Lecture Notes in Computer Science, 2009; 5574: 693-703.

172. Muppavarapu V, Chung SM. Semantic-based access control for grid data resources in Open Grid Services Architecture - Data Access and Integration (OGSA-DAI). En Actas: International Conference on Tools with Artificial Intelligence (ICTAI), 2008.
173. Kolter J, Schillinger R, Pernul G. Building a distributed semantic-aware security architecture. IFIP International Federation for Information Processing, 2007; 232: 397-408.
174. Rahimi B, Vimarlund V. Methods to evaluate health information systems in healthcare settings: A literature review. J Med Syst, 2007; 31(5): 397-432.
175. MacLean A, Young RM, Bellotti VME, Moran TP. Questions, options, and criteria. Elements of design space analysis. Hum-Comput Interact, 1991; 6(3-4): 201-250.
176. Emiliani PL, Stephanidis C. Universal access to ambient intelligence environments: Opportunities and challenges for people with disabilities. IBM Syst J, 2005; 44(3): 605-619.
177. Yahiaoui N, Traverson B, Levy N. A new viewpoint for change management in RM-ODP systems. En Actas: 2nd International Workshop on ODP for Enterprise Computing (WODPEC), 2005: 1-6.
178. Sottile PA. Approaches in European health information systems architectures. Stud Health Technol Inform, 2003; 96: 29-37.
179. Román I, Roa LM, Madinabeitia G, Millán A. Introducing guideline management in the healthcare information system architecture. Stud Health Technol Inform, 2007; 127: 117-124.
180. Senivongse T, Teng-amnuay Y, Nupairoj N. A Comparison of System Modelling for Distributed Applications: RM-ODP vs MDA.
181. Putman J. Architecting with RM-ODP, 2001.
182. Gervais MP. Towards an MDA-oriented methodology. In Proceedings - IEEE Computer Society's International Computer Software and Applications Conference, 2002.
183. Traverson BA. Comparison Study of Viewpoint Approaches in Service Enterprise Architecture. En Actas: 12th Enterprise Distributed Object Computing Conference Workshops, 2008.
184. Romero JR, Jaén JI, Vallecillo A. Realizing correspondences in multi-viewpoint specifications. En Actas: 13th IEEE International Enterprise Distributed Object Computing Conference (EDOC), 2009.
185. Massacci F, Zannone N. A model-driven approach for the specification and analysis of access control policies. Lecture Notes in Computer Science, 2008; 5332: 1087-1103.
186. Lodderstedt T, Basin D, Doser J. SecureUML: A UML-based modeling language for model-driven security. Lecture Notes in Computer Science, 2002; 2460: 426-441.
187. Miyazaki H, Tanaka A. Study on representation of security aspects in each viewpoint using UML for ODP. En Actas: IEEE International Enterprise Distributed Object Computing Workshop (EDOC), 2007.
188. Basin D, Doser J, Lodderstedt T. Model driven security: From UML models to access control infrastructures. ACM Transactions on Software Engineering and Methodology, 2006; 15(1): 39-91.
189. Román I. Aportaciones Metodológicas a la Integración de Sistemas de Información Sanitarios basada en Gestión Semántica, 2006.
190. Jena Community. Jena Semantic Web Framework. Disponible en: http://jena.sourceforge.net/.
191. Ekdahl AW, Andersson L, Friedrichsen M. "They do what they think is the best for me." Frail elderly patients' preferences for participation in their care during hospitalization. Patient Educ Couns, 2010; 80(2): 233-240.
192. Haux R. Individualization, globalization and health - about sustainable information technologies and the aim of medical informatics. Int J Med Inf, 2006; 75(12): 795-808.
193. US Government. Health Insurance Portability and Accountability Act (HIPAA), 1996.

194. International Medical Informatics Association. Code of ethics for health information professionals, 2002.
195. Council of Europe. Convention for the protection of human rights and dignity of the human being with regard to the application of biology and medicine, 1997.
196. World Health Organization. A declaration on the promotion of patients' rights in Europe, 1994: 10-17.
197. European Commission. Directive 95/46/EC of the European parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data, 1995.
198. Markle Foundation. Connecting for health: The personal health working group final report, 2003.
199. Bourgeois FC, Taylor PL, Emans SJ, Nigrin DJ, Mandl KD. Whose Personal Control? Creating Private, Personally Controlled Health Records for Pediatric and Adolescent Patients. Journal of the American Medical Informatics Association, 2008; 15(6): 737-743.
200. Aubert BA, Hamel G. Adoption of smart cards in the medical sector: The Canadian experience. Social Science and Medicine, 2001; 53(7): 879-894.
201. Maloney FL, Wright A. USB-based Personal Health Records: An analysis of features and functionality. Int J Med Inf, 2010; 79(2): 97-111.
202. Reti SR, Feldman HJ, Safran C. Governance for Personal Health Records. Journal of the American Medical Informatics Association, 2009; 16(1): 14-17.
203. Sinnott RO, Chadwick DW, Doherty T, Martin D, Stell A, Stewart G, Su L, Watt J. Advanced security for virtual organizations: The pros and cons of centralized vs decentralized security models, 2008. En Actas: 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID), 2008, 106-113.
204. Kerschbaum F, Robinson P. Security architecture for virtual organizations of business web services. J Syst Archit, 2009; 55(4): 224-232.
205. Zhang N, Yao L, Nenadic A, Chin J, Goble C, Rector A, et al. Achieving fine-grained access control in virtual organizations. Concurrency Computation Practice and Experience, 2007; 19(9): 1333-1352.
206. Rector AL, Nowlan WA, Consortium G. The Galen project. Comput Methods Programs Biomed, 1994; 45(1-2): 75-78.
207. OBO Foundry. Basic Formal Ontology. Disponible en: http://obofoundry.org/cgi-bin/detail.cgi?bfo.
208. Smith B, Ashburner M, Rosse C, Bard J, Bug W, Ceusters W, et al. The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. Nat Biotechnol, 2007; 25(11): 1251-1255.
209. Eclipse. Eclipse Modeling Framework (EMF). Disponible en: http://www.eclipse.org/emf/.
210. Amyot D, Farah H, Roy J. Evaluation of development tools for domain-specific modeling languages. Lecture Notes in Computer Science, 2006; 4320: 183-197.
211. Eclipse. Graphical Modeling Framework (GMF). Disponible en: http://www.eclipse.org/gmf/.
212. Kolovos DS, Rose LM, Paige RF, Polack FAC. Raising the level of abstraction in the development of GMF-based graphical model editors. En Actas: International Conference on Software Engineering, 2009.
213. Eclipse. MOFScript. Disponible en: http://www.eclipse.org/gmt/mofscript/.
214. Eclipse. Model-To-Text initiative. Disponible en: http://www.eclipse.org/modeling/m2t/.
215. Eclipse. Emfatic Language Reference. Disponible en: http://www.eclipse.org/gmt/epsilon/doc/articles/emfatic/.
216. Eclipse. Epsilon Validation Language (EVL). Disponible en: http://www.eclipse.org/gmt/epsilon/doc/evl/.
217. ITU-T Rec. X.500 The directory: Overview of concepts, models and services, 2008.
218. OpenLDAP community. The OpenLDAP. Disponible en: http://www.openldap.org/.
219. PERMIS Home. Disponible en: http://www.permis.org/index.html.

220. ISO. ISO 21091: Health informatics - Directory services for security, communications and identification of professionals and patients, 2009.
221. Tools for HISA specification, 2012. Disponible en: http://gibserv.us.es/wiki/doku.php.
222. Calvillo J, Román I, Rivas S, Roa LM. Easing the development of healthcare architectures following RM-ODP principles and healthcare standards. Computer Standards and Interfaces (pendiente de publicación impresa). (10.1016/j.csi.2011.12.002), 2012.
223. Calvillo J, Román I, Rivas S, Roa LM. Privilege management infrastructure for virtual organizations in healthcare grids. IEEE Transactions on Information Technology in Biomedicine, 2011; 15(2): 316-323.
224. Calvillo J, Román I, Roa LM. Empowering citizens with access control mechanisms to their personal health resources. Int J Med Inf (pendiente de publicación impresa), (10.1016/j.ijmedinf.2012.02.006), 2012.

Anexo. Aportaciones científicas

1. Introducción

En este capítulo se describen las aportaciones científicas presentes en esta Tesis Doctoral y los trabajos científicos que ha dado lugar su desarrollo. La necesidad de investigar y diseñar nuevos escenarios de asistencia socio-sanitaria centrados en el ciudadano y con base en arquitecturas normalizadas y abiertas ha llevado a trabajar en varias disciplinas de conocimiento como son los sistemas distribuidos, los métodos de formalización de arquitecturas, los mecanismos de seguridad, la gestión de semántica y el rol del SdA en la gestión de sus recursos sanitarios. Los objetivos particulares del trabajo en cada una de estas áreas eran:

1. la armonización de los mecanismos y estándares relacionados con la autorización y el control de acceso y su aplicación a la distribución de componentes en el dominio sanitario;

2. la reestructuración de la arquitectura definida en la norma ISO12967 para mejorar su conformidad con el modelo de referencia ODP;

3. la extensión de la arquitectura para cubrir la necesidad de control de acceso y gestión de privilegios, facilitando la integración de estos servicios transversales con el resto de la organización, y la aplicación de fundamentos semánticos (ontologías, motores de inferencia, etc.) al control de acceso; y

4. el diseño de un modelo que soporte la autonomía del SdA con éste como administrador de sus recursos y con capacidad para otorgar privilegios de acceso.

Aunque a priori podría parecer que dichos objetivos son independientes entre sí todos están relacionados a través del objetivo general de esta Tesis Doctoral. En la Figura 5.2 puede verse cómo los esfuerzos realizados para alcanzar dichos objetivos se relacionan y contribuyen al desarrollo del escenario POVO. A continuación se realiza una breve descripción de dichas contribuciones y se explica por qué y en qué medida son originales.

2. Contribuciones originales

El primer punto abordado en este trabajo ha sido la formalización de uno de los esfuerzos más relevantes para el desarrollo de arquitecturas normalizadas aplicadas al dominio sanitario, el estándar HISA. Los requisitos y las especificaciones de esta norma han sido formalizados conforme a RM-ODP, su lenguaje de empresa y el perfil UML4ODP lo que implica numerosas aportaciones a este campo de conocimiento. La más evidente es que los modelos desarrollados permiten que el estándar HISA pueda ser llevado a la práctica por diseñadores y desarrolladores ya que se dispone de una representación formal y reutilizable del esqueleto de la arquitectura normalizada. Para potenciar el impacto de esta contribución las herramientas desarrolladas, es decir, los tres puntos de vista independientes de tecnología de HISA conforme a RM-ODP, el lenguaje de empresa y UML4ODP, están disponibles como material de apoyo para los interesados en la construcción de soluciones siguiendo los principios de estas

normas. Se ha habilitado una página web [221] y se han publicado estas contribuciones con el fin de extender su difusión [222].

Por otro lado se han identificado afinidades y divergencias entre HISA y RM-ODP así como puntos fuertes y débiles de la norma sanitaria. Estos resultados constituyen un interesante compendio de información de entrada para la evolución potencial del estándar HISA así como para facilitar su entendimiento y relaciones con RM-ODP u otros estándares. En la Figura A.1 se muestra cómo influye esta contribución con el paradigma POVO de asistencia socio-sanitaria centrada en el SdA. Tanto el paradigma de diseño SOA como la especificación de HISA conforme a RM-ODP y UML4ODP serán los fundamentos para la formalización y diseño de la arquitectura del paradigma POVO.

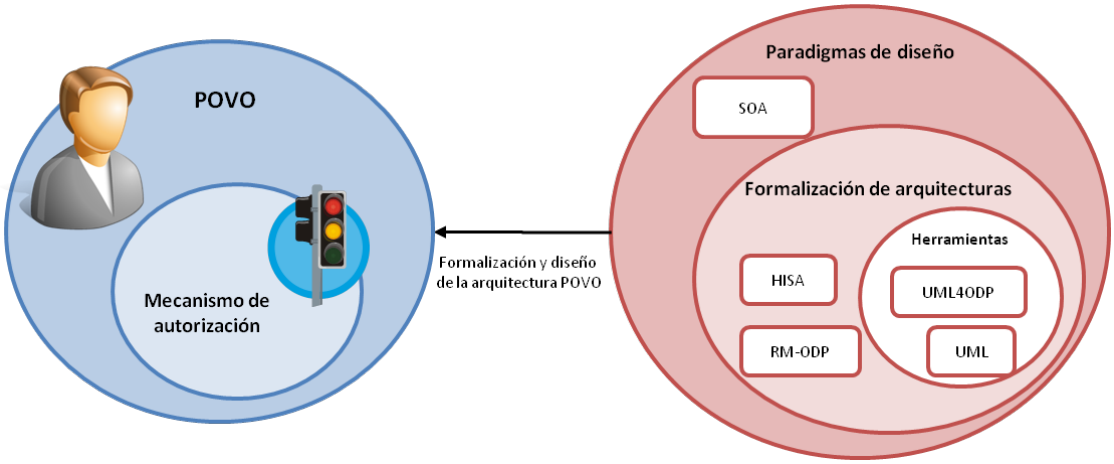

Figura A.1. Relación de la arquitectura normalizada HISA con el paradigma POVO

El segundo aspecto en el que ha incidido este trabajo ha sido el establecimiento de una armonización de estándares y normativa referentes a la seguridad en general y el control de acceso en particular. Esto se debe a que el escenario POVO requiere mecanismos avanzados de autorización que contemplen una enorme heterogeneidad y distribución de recursos, dominios y usuarios. Además la introducción de mecanismos de seguridad en una arquitectura normalizada requería de una armonización de conceptos si no se quería perder la apertura y estandarización de dicha arquitectura. Se ha establecido, por tanto, un marco conceptual común a las distintas iniciativas y estándares relevantes en este campo a partir del mapeo de la semántica de sus conceptos. En la Figura A.2 se muestra la relación entre este esquema normalizado y el mecanismo de autorización del paradigma POVO. Esta armonización de iniciativas de control de acceso aplicada a escenarios concretos ha sido publicada en [223].

La armonización de conceptos ha dado lugar a un esquema de control de acceso basado en atributos suficientemente flexible como para dar cabida a distintos tipos de escenarios, mecanismos de autorización e iniciativas normalizadas. Así por ejemplo, mientras que en un escenario centralizado todos los elementos estarán dentro del mismo dominio de seguridad tanto física como lógicamente y cada uno tendrá constancia de cómo comunicarse con el resto, en un escenario distribuido los elementos pueden estar localizados en distintos dominios e incluso existir multiplicidad de elementos

(varios agentes de decisión, por ejemplo). Teniendo en cuenta que el escenario POVO es fuertemente distribuido se ha supuesto transparencia de distribución y localización, es decir, no se pretende aquí dar una resolución al descubrimiento y localización de elementos. Así el esquema de autorización presentado es adaptable a un escenario distribuido multi-dominio considerando que existe una capa inferior que proporciona la resolución de distribución y localización. A ella se tendrán que dirigir los elementos para encontrar a sus interlocutores más adecuados en cada caso y proporcionará además características como balanceo de carga y replicación de elementos.

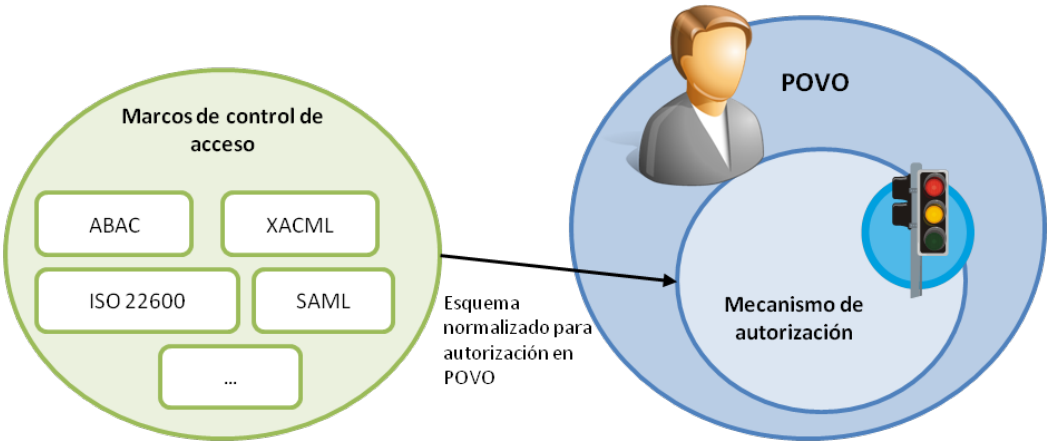

Figura A.2. Relación entre la armonización de marcos de autorización y el paradigma POVO

El esquema de control de acceso obtenido no se utiliza de forma independiente sobre el escenario POVO sino que, en un paso previo, se integra formalmente con la arquitectura sanitaria normalizada basada en HISA. Para ello las tres vistas independientes de tecnología para el sistema de seguridad del paradigma POVO han sido desarrolladas como un refinamiento y extensión de la arquitectura normalizada de HISA considerando sus procedimientos normativos así como los principios de RM-ODP, el lenguaje de empresa y el perfil UML4ODP. Las implementaciones y soluciones tecnológicas del mecanismo de autorización se han dejado para las vistas dependientes de tecnología de la arquitectura (ver más adelante).

Uno de los principales beneficios de esta contribución es la posibilidad de ser reutilizados por otros diseñadores y desarrolladores y servir como material de apoyo para los interesados en la construcción de soluciones siguiendo los principios de RM-ODP e HISA. Para facilitar su reutilización, las herramientas desarrolladas (los tres puntos de vista independientes de tecnología del sistema de autorización conforme a HISA y RM-ODP) están disponibles vía web [221] y se han publicado con el fin de extender su difusión [222-224]. Este esfuerzo puede además ser considerado como una prueba de la usabilidad de la formalización de HISA así como de los procedimientos de extensión de la norma ISO12967. En la Figura A.3 se muestra cómo este resultado permite extender la especificación de la norma HISA con el marco armonizado de control de acceso.

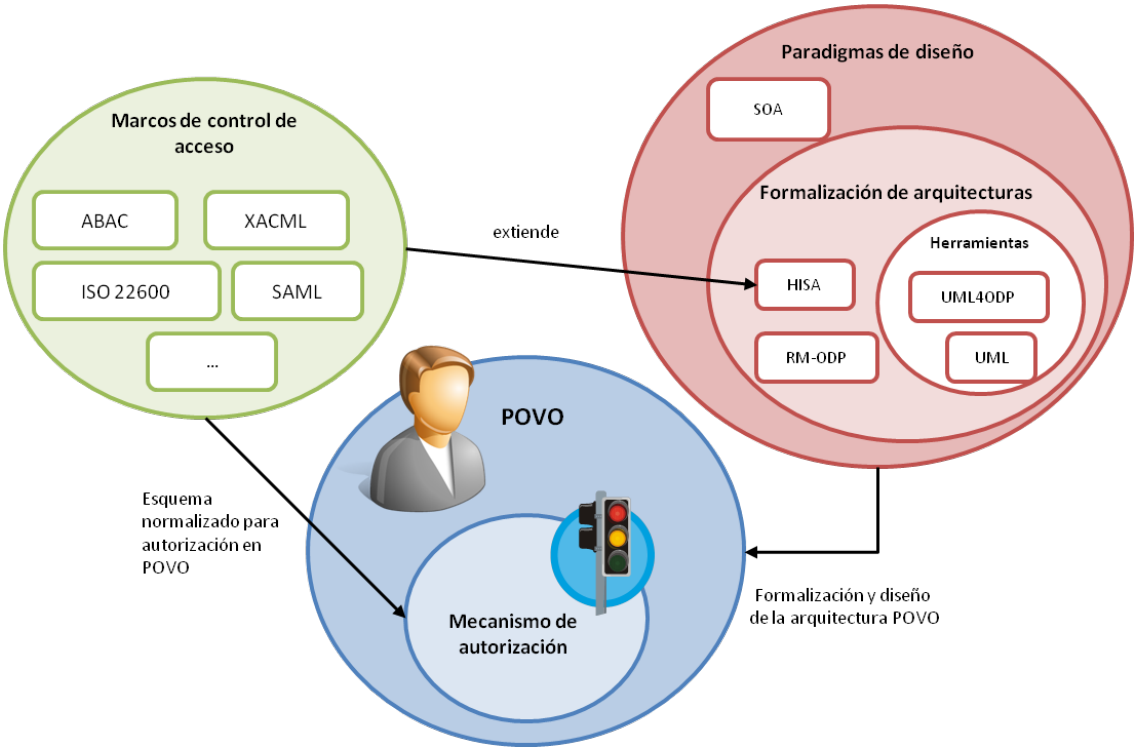

Figura A.3. Relación entre el marco armonizado de autorización y la especificación de HISA

Otro aspecto elemental que necesita ser considerado en el paradigma POVO es la necesidad de aunar elementos distribuidos a través de diferentes dominios administrativos y tecnológicos lo que se traduce en un aumento de la complejidad de la solución. Buscando potenciar la interoperatividad y apertura de la arquitectura se han centrado los esfuerzos en la definición formal de un sistema de gestión de elementos semánticos transversal a todos los servicios de la arquitectura. Este sistema, al igual que se hizo con el mecanismo de autorización, se ha especificado como un refinamiento y extensión de la arquitectura normalizada de HISA (Figura A.4) considerando sus procedimientos normativos así como los principios de RM-ODP, el lenguaje de empresa y el perfil UML4ODP.

Al igual que ocurre con las contribuciones anteriores, la especificación del sistema de gestión de semántica puede ser reutilizada por diseñadores y desarrolladores, sirviendo como material de apoyo para los interesados en la construcción de soluciones con base semántica y siguiendo los principios de RM-ODP e HISA. Dentro del paradigma POVO y teniendo en cuenta el alcance de esta Tesis Doctoral, la gestión de semántica se ha utilizado para otorgar al mecanismo de control de acceso capacidades semánticas que facilitan su operación entre dominios ontológicos y la automatización de algunos de sus procesos. Paralelamente, esta contribución tiene un gran potencial a nivel de la norma HISA ya que en su versión actual cubre muy levemente la gestión de elementos semánticos. Este aspecto es clave en las arquitecturas de sistemas distribuidos ya que es una capacidad transversal a todos los sistemas y merece ser desarrollado en profundidad. Por todo ello, el trabajo realizado aquí puede servir de realimentación para futuras revisiones de la norma ISO12967.

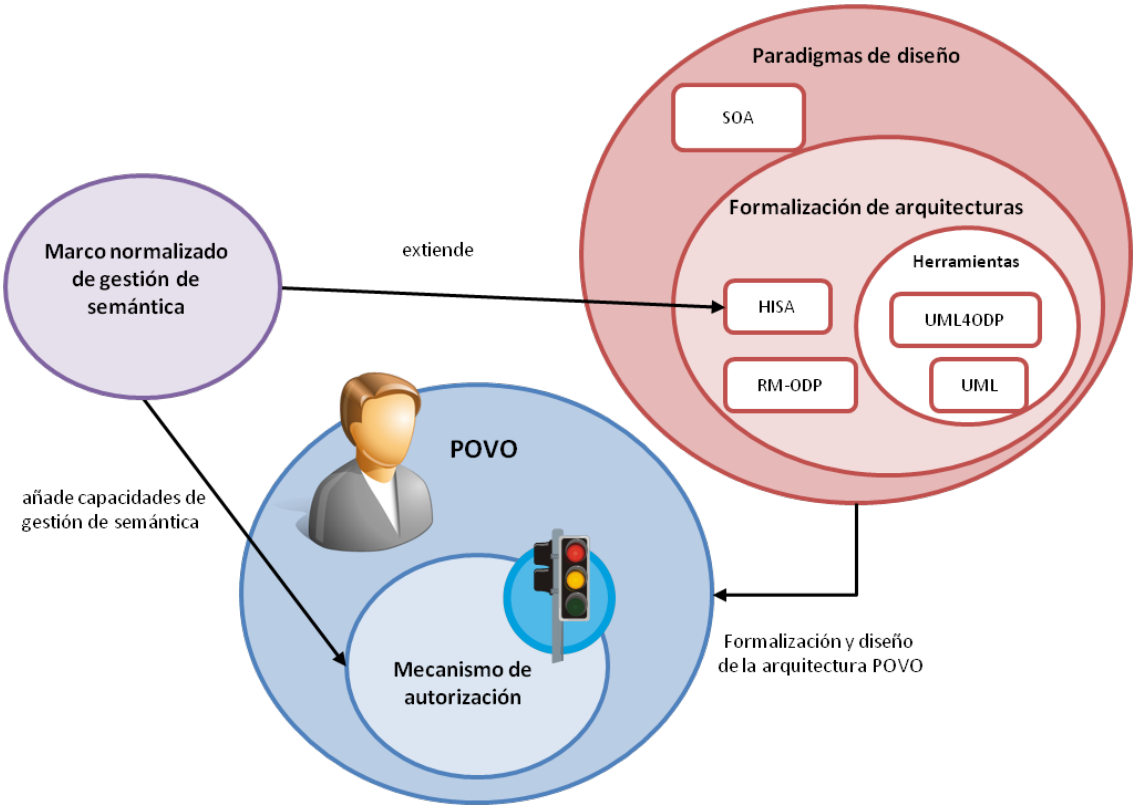

Figura A.4. Relación entre el marco normalizado de gestión de semántica y la especificación de HISA

La última contribución del presente trabajo de Tesis Doctoral es el objetivo principal de la misma, esto es, el diseño y desarrollo del escenario POVO como paradigma de sistema sanitario centrado en el SdA. Para diseñar el paradigma POVO y desarrollar algunos aspectos concretos de su infraestructura se han reutilizado los resultados de las anteriores aportaciones. Por un lado, y siguiendo la metodología utilizada en todo el trabajo, se han combinado las especificaciones independientes de tecnología de la arquitectura sanitaria basada en HISA y sus extensiones de seguridad y gestión de semántica como base de la arquitectura de soporte para la POVO. Esta arquitectura sanitaria normalizada se ha particularizado para el paradigma de diseño SOA y las tecnologías de computación en Grid a través de la especificación de los puntos de vista de Ingeniería y Tecnología. Por otro lado, como muestra de la potencialidad del paradigma POVO, se ha desarrollado un mecanismo de control de acceso basado en semántica y con usabilidad suficiente para ser utilizado por cualquier SdA. Para ello se ha hecho uso de las capacidades de gestión de semántica incorporadas a la arquitectura y de diversas tecnologías semánticas para la construcción del agente que toma las decisiones de control de acceso, el editor de políticas y su traducción a lenguaje procesable, y las ontologías de conceptos que potencian la interoperatividad y automatización de los procesos. Todas estas contribuciones aparecen publicadas en [224]. En la Figura 5.2 se muestran todas las contribuciones del presente trabajo de Tesis Doctoral y cómo se relacionan entre sí para construir el paradigma POVO y su mecanismo de control de acceso.

3. Publicaciones científicas

La investigación realizada durante el desarrollo de esta Tesis Doctoral ha dado lugar a las siguientes publicaciones.

3.1. Año 2012

3.1.1. Publicaciones en revistas

Empowering citizens with access control mechanisms to their personal health resources

J. Calvillo, I. Román and L.M. Roa

International Journal of Medical Informatics

Pendiente de publicación

DOI: 10.1016/j.ijmedinf.2012.02.006

Easing the development of healthcare architectures following RM-ODP principles and healthcare standards

J. Calvillo, I. Román, S. Rivas and L.M. Roa

Computer Standards & Interfaces

Pendiente de publicación

DOI: 10.1016/j.csi.2011.12.002

How technology is empowering patients? A literature review

J. Calvillo, I. Román and L.M. Roa

Health Expectations

En proceso de revisión

3.1.2. Conferencias impartidas

Normalized architectures supporting person-centric healthcare delivery

Interconnected Health 2012

Rosemont, Illinois, USA

Telemedicina y Salud. Un caso de estudio: el proyecto e-Nefro

Seminario 'TIC al Servicio de la Sociedad 2012'

Escuela Superior de Ingeniería, Universidad de Sevilla

3.2. Año 2011

3.2.1. Publicaciones en revistas

Privilege management infrastructure for virtual organizations in healthcare grids

J. Calvillo, I. Román, S. Rivas and L.M. Roa

IEEE transactions on information technology in biomedicine, 15 (2), pp 316-23

3.2.2. Aportaciones a congresos

Herramientas semánticas orientadas al ciudadano para el control de acceso a sus recursos sanitarios

J. Calvillo, I. Román and L.M. Roa

XXIX Congreso Anual de la Sociedad Española de Ingeniería Biomédica, Actas, pp. 151-4

3.3. Año 2010

3.3.1. Capítulos de libro

A comprehensive view of the technologies involved in pervasive care

L.M. Roa, L.J. Reina, M.A. Estudillo, J. Calvillo and I. Román

Future visions on biomedicine and bioinformatics, Volume 1, pp. 3-19

Editorial: Springer-Verlag

ISBN: 978-3-642-15050-0

3.4. Año 2009

3.4.1. Capítulos de libro

Personalizing care: integration of hospital and homecare

I. Román, J. Calvillo and L.M. Roa

Handbook of digital homecare, Biomed. pp. 33-52

Editorial: Springer-Verlag

ISBN: 978-3-642-01386-7

3.4.2. Aportaciones a congresos

Privilege management infrastructure for virtual organizations in healthcare grids

I. Román, J. Calvillo, S. Rivas and L.M. Roa

International Conference on Information Technology and Applications in Biomedicine, ITAB 2009, Actas, pp. 1-4

Infraestructuras de seguridad para arquitecturas de servicios sanitarios distribuidos

I. Román, J. Calvillo, S. Rivas and L.M. Roa

XXVII Congreso Anual de la Sociedad Española de Ingeniería Biomédica, Actas, pp. 697-700.

Servicio web para la gestión de sesiones clínicas en un servicio de cirugía plástica

S. Rivas, I. Román, J. Calvillo, P.T. Gómez and L.M. Roa

XXVII Congreso Anual de la Sociedad Española de Ingeniería Biomédica, Actas, pp. 361-4.

3.5. Año 2008

3.5.1. Capítulos de libro

Improving healthcare middleware standards with semantic methods and technologies

I. Román, J. Calvillo, L.M. Roa and G. Madinabeitia

Medical and Care Compunetics 5, pp. 181-9.

Editorial: IOS PRESS

ISBN: 978-1-58603-868-7

3.5.2. Aportaciones a congresos

Servicios para la gestión y evaluación de guías clínicas en formato XML

I. Román, J. Calvillo, L.M. Roa, C. Maya, F. García

XXVI Congreso Anual de la Sociedad Española de Ingeniería Biomédica, Actas, pp. 403-6

3.6. Aportaciones a procesos normativos

Paralelamente a las publicaciones, el ámbito de la investigación ha propiciado el contacto con grupos de estandarización nacionales e internacionales y contribuciones al desarrollo o revisión de diversas normas.

- ISO 14292 Personal Health Records: Definition, Scope and Context, 2012.
- CEN/TS 15699 Clinical knowledge resources – Metadata, 2009.
- ISO 21091: Health informatics - Directory services for security, communications and identification of professionals and patients, 2005. (Revisión 2009)
- ISO 18308 Health Informatics - Requirements for an electronic health record reference architecture, 2009.